Provably Secure Authenticated Encryption Modes

Student Name: R Sumesh Manjunath

IIIT-D-MTech-CS-IS-11-013
May 16, 2013

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Somitra Kumar Sanadhya (Chair)
Ragesh Jaiswal
Gaurav Gupta

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science,
with specialization in Information Security

Keywords: Authenticated Encryption, Game Playing Framework, Provable Security, Privacy, Authenticity, FWPAE, FPAE

Certificate

This is to certify that the thesis titled **"Provably Secure Authenticated Encryption Modes"** submitted by **R Sumesh Manjunath** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by him under our guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the award of any other degree.

Prof. Somitra Kumar Sanadhya
IIIT Delhi

Prof. Donghoon Chang
IIIT Delhi

Abstract

Privacy of the message and authenticity of the sender in a secure communication is a challenging concern. Tradionally these two aims were achieved by using different cryptographic primitives: by using encryption for privacy and using MAC's for authenticity. Authenticated Encryption (AE) is a mechanism to provides both the privacy of data as well as authenticity of the sender by a single cryptographic construction. Usually, AE schemes have been constructed as mode of operation of a block cipher providing both confidentiality and authenticity.

Bellare and Namprempre [1] introduced the idea of AE and showed different compositions of Encryption and MAC schemes to construct AE schemes, along with the security proof for each construction. In their work, Bellare and Namprempre also highlighted the subtle issues which can lead to insecurity in some combinations of encryption and MAC schemes.

Many modes of AE have been developed after the pioneering work of Bellare et al. in 2000. Jutla developed the IAPM [7] mode in 2001. Around the same time, Rogaway et al. proposed the OCB mode [15]. OCB is one of the most efficient AE modes. Other efficient modes are CCM [16] and CWC [9]. All of these AE modes are based on block ciphers. The SpongeWrap [4] is the only known AE scheme based on a permutation, while there is no known AE mode which is based on a random function. Near lack of non-block cipher based designs for AE motivated us to study new AE designs. In this work, we propose two new AE modes. The first one, which we name FWPAE is based on random function and the second one, which we call FPAE, is based on permutation. Our proposed permutation based mode FPAE promises to have better security compared to SpongeWrap.

Acknowledgments

Mata Pita Guru Dev (Mother Father Teacher Lord)

I dedicate this thesis work to my parents B.G Ramesh and R. Jayalakshmi. Without their love and support I'm nothing. Thank you mom and dad for being so supportive.

It is always a pleasure to thank every person who helped me in the completion of this work.

First, I sincerely thank Dr. Somitra Kumar Sanadhya who introduced me to cryptography, without him I would not have started my career in cryptography. I will always be grateful to you throughout my life sir. Every conversation is inspiring and gives new idea to proceed further. You have never said he doesn't have time for the discussion. You are an inspiration for me sir.

Next person I must thank is Dr. Donghoon Chang. He is the one who exposed me to this area and helped me in every step I progress. Every discussion helped us in finding new research problems to work on. Your energy and enthusiasm is very inspiring sir.

I thank Dr. Debajyoti Bera for giving us an inspiring class on Theory of Modern Cryptography which helped me to proceed faster for my thesis. Your way of teaching is is very helpful to get a very clear insight about the topic being taught.

I thank Dr. Pankaj Jalote for creating IIIT Delhi and giving me an opportunity to do my master program. I thank the office people especially Mr. VIvek Tiwari for being very supportive from administrative side.

Friends always play an important role in every stage of our life and I'm not an exception. I take this opportunity to thank everybody who had been part of this phase especially, Abhishek Kumar, Mohona Ghosh, Tarun, Amit for fruitful discussion on cryptography and Pandarasamy Arjunan, Jayaprakash, Anush Sankaran, Tejas, Dipto, Robin, Monalisa, Sweta Mishra, Monika Gupta, Amani, Vidushi, Piyush, Vivekanand and many people with whom I had fruitful, fun and sometimes non technical discussion.

Last but not least, I thank Dr. Ragesh Jaiswal from IIT-Delhi, Dr. Gaurav Gupta from IIIT-Delhi for accepting to be a committee member for my thesis defense.

R. Sumesh Manjunath, May 2013

Contents

1 Introduction **1**

1.1 Basic Setting of Cryptography 1

1.2 Symmetric Key Cryptography 2

1.2.1 Privacy : Encryption Scheme 2

1.2.2 Authenticity : Message Authentication Code 3

2 Authenticated Encryption **4**

2.1 What is Authenticated Encryption 4

2.2 Definition of Authenticated Encryption 4

2.3 Modes of Authenticated Encryption 5

2.3.1 Basic combinations 5

2.3.2 Integrity Aware Parallelizable Mode 6

2.3.3 Offset CodeBook Mode 7

2.3.4 SpongeWrap 7

3 Preliminaries **9**

3.1 Provable Security 9

3.2 Adversary Models 9

3.2.1 Standard Model 10

3.2.2 Random Oracle Model 10

3.2.3 Ideal Cipher Model 10

3.3 Random Oracle 10

3.4 Ideal Permutation 11

3.5 Security Notions for Authenticated Encryption 11

3.5.1 Privacy 11

3.5.2 Authenticity 12

3.6 Frameworks 13

3.6.1 Game Playing Framework 13

3.6.2 Indifferentiability Framework 14

4 FWPAE **15**
4.1 Description . . . 15
4.2 FWP Mode . . . 15
4.3 Notations . . . 16
4.4 Structure . . . 16
4.5 Encryption Algorithm . . . 16
4.6 Decryption Algorithm . . . 17
4.7 Security Proofs . . . 18
4.7.1 Privacy . . . 18
4.7.2 Authenticity . . . 21
4.8 Games . . . 25

5 FPAE **30**
5.1 Description . . . 30
5.2 FP Mode . . . 30
5.3 Structure . . . 30
5.4 Encryption Algorithm . . . 31
5.5 Decryption Algorithm . . . 31
5.6 Security Proofs . . . 32
5.6.1 Privacy . . . 32
5.6.2 Authenticity . . . 37
5.7 Games . . . 41

6 Conclusion **48**

7 Future Work **49**

List of Figures

1.1 Basic cryptographic setting 2

2.1 Encrypt-and-Authenticate 5
2.2 Authenticate-then-Encrypt 6
2.3 Encrypt-then-Authenticate 6
2.4 IAPM Mode 7
2.5 OCB Mode 7
2.6 SpongeWrap 8

3.1 Privacy:Indistinguishability. Adversary has to find out whether it is interacting with AE scheme or a random oracle. 12
3.2 Authenticity:forgery. Adversary has to forge ciphertext, tag pair. Adversary has access to encryption, decryption and primitive oracles to forge. 13

4.1 FWPAE: Encryption 16

5.1 FPAE: Encryption 31

List of Tables

4.1 FWPAE: Notations 16
4.2 FWPAE: Symbol descriptions 16
4.3 FWPAE: Game G0 25
4.4 FWPAE: Game G1 & G2 26
4.5 FWPAE: Game G3 & G4 27
4.6 FWPAE: Game G5 28
4.7 FWPAE: Game G6 28
4.8 FWPAE: Game G7 29

5.1 FPAE: Symbol Descriptions 31
5.2 FPAE: Game G0 42
5.3 FPAE: Game G1 & G2 43
5.4 FPAE: Game G3 & G4 44
5.5 FPAE: Game G5 & G6 45
5.6 FPAE: Game G7 46
5.7 FPAE: Game G8 47

Chapter 1

Introduction

1.1 Basic Setting of Cryptography

Secrecy of exchanged messages between communicating parties has been a desired feature for some since time immemorial. In the older times, only the governments and armies had a need for this. However, with the advent of digital world, it has become a requirement for a common person. Both the privacy of the communication as well its authenticity are required for anyone who, for example, accesses his mails stored on the servers of the servide providers, or one who accesses his bank details online. Traditional cryptography, which was the only type of cryptography ussed till the 80's, was concerned about designing tools which make it "difficult" for the attacker to make sense of ciphered data. Modern cryptography takes the notion of security to a different plane, by first rigorously defining what is meant by the security of the scheme and then concretely quantifying the security [8].

The basic setting of cryptography is explained better with few characters depicting the real world scenario. Consider a two-party communication over an *open* channel (here *open* means the vulnerability to leakage of data). A sender, S, who sends information over an open channel and a receiver, R, who receives the information from the sender over this channel. The vulnerability here means the possibility of data being read and/or modified and this scenario is depicted by a new character Adversary, A. The power of the adversary can be modeled ranging from only reading the data (passive adversary) to modifying the information in the open channel (active adversary).

Since the passive adversary can read the data in the open channel, there must be some encoding, decoding functions with some secret information so that the adversary cannot find the actual information being transfered. This secret information is called the 'Key', the encoding and decoding functions are called 'Encryption' and 'Decryption' algorithms, respectively. This basic setting is depicted in the figure:1.1. The 'Key' plays an important role in cryptography. Based on the type of key sharing between the two legitimate parties, cryptography can be divided into the following two main branches, *a*) Symmetric Key Cryptography, *b*) Asymmetric Key Cryptography.

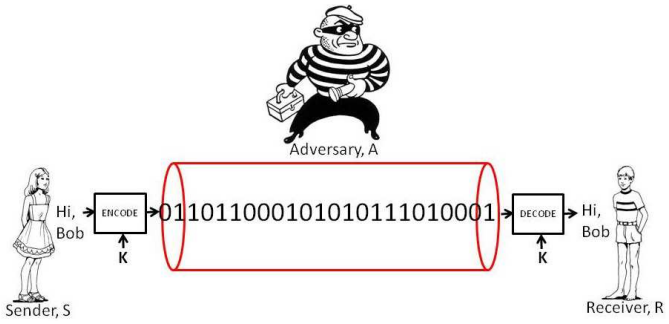

Figure 1.1: Basic cryptographic setting

1.2 Symmetric Key Cryptography

The Key used by S and R are the same. All the symmetric Cryptography schemes and protocols are developed based on some assumptions such as *a*) The shared secret key is already present between sender and receiver; *b*) Adversary cannot get the key from sender or receiver by any means. In the symmetric key setting, privacy of a message is provided by an Encryption scheme and Message authenticity by Message Authentication Code (MAC). These two are the basic properties which need to be satisfied for communication to take place in a open channel. We will see these two briefly.

1.2.1 Privacy : Encryption Scheme

An encryption scheme in symmetric setting is defined as consisting of 3 tuples $\Pi^{'} = (\mathcal{K}^{'}, \mathcal{E}^{'}, \mathcal{D}^{'})$ where the notation is defined ahead shortly. The information which is desired to be sent by the sender is called *plaintext*, and the encoded data is called *ciphertext*. The encoding function is called Encryption algorithm and is represented by $\mathcal{E}^{'}$. This algorithm takes plaintext and the shared key as input and outputs the ciphertext. In the same way, the decoding function is called the Decryption algorithm and is denoted by $\mathcal{D}^{'}$. This algorithm takes the ciphertext received through the open channel and the shared key as input and outputs the plaintext. The algorithm $\mathcal{K}^{'}$ is called the key generation algorithm. The purpose of this algorithm is to produce a shared key which is used by both $\mathcal{E}^{'}, \mathcal{D}^{'}$. Usually, this algorithm is defined as one which produces a random string of the required key length.

The goal of an encryption algorithm is to make sure that the adversary cannot gain any information about the plaintext from the ciphertext (other than the length of the plaintext). We exclude the lenngth information, since the length of plaintext can usually be calculated from the length of ciphertext.

From computational point of view, the security of a scheme is defined in terms of the probability to break some security notion relating to the scheme. As mentioned before, only the key is kept secret from the adversary. If the encryption scheme has a key length of n bits, then with probability 2^{-n}, the adversary can find the correct key in a single attempt. Once the key is

known, then the adversary can decrypt the ciphertext as if he was the legitimate receipient.

1.2.2 Authenticity : Message Authentication Code

Since the transmission channel is assumed to be insecure, the receiver must be able to identify whether the data it received is from the correct sender and not modified by any adversary in the channel.

Message Authentication Code (MAC) is used to provide authenticity. MAC scheme is a 3-tuple of algorithms denoted by $\Pi'' = (\mathcal{K}'', \mathcal{T}, \mathcal{V})$. Algorithm $\mathcal{T}$ is a function which takes a message and the shared key as input and outputs a small string called Tag (σ) which is then transmitted to the receiver as an authenticator. This tag cannot be calculated without knowing the key. Once the tag is received, the receiver uses the algorithm $\mathcal{V}$ using the key and the message to verify that the received tag is correct. If the algorithm returns 1, then the message is authenticate otherwise, the message has been tampered with.

Chapter 2

Authenticated Encryption

2.1 What is Authenticated Encryption

Privacy and authenticity are the two separate basic security goals and there are many schemes which provide both separately. There are many applications which require both features to be present in a scheme. It is not so difficult to see that for any message which is sent through open channel both the privacy as well as authenticity are required in a single scheme. Bellare et. al defined a notion for this type of scheme as Authenticated Encryption(referred to as AE at many places from now on) in [1]. AE provides both privacy as well as authenticity.

There could be a scenario where the data sent passes through many different channels. Thus some part of the data, say header in a mail message, must be in plaintext for proper delivery of the data. But even in this case, the scheme must make sure that the whole data including the header is not modified. In cryptography, this scenario can be said as a partial encryption and complete authentication of data and we use the term *Authenticated Encryption with Associated Data*, shortly AEAD, for it.

2.2 Definition of Authenticated Encryption

Authenticated Encryption has been informally introduced in the previous section. Now we formally define AE. An AE scheme is defined as $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. $\mathcal{K}$ is the key generating algorithm. Usually a random string, K of key length is produced as its output. $\mathcal{E}$ is called encryption algorithm which takes K and plaintext, optionally a nonce as input, and outputs ciphertext and tag. Decryption algorithm, $\mathcal{D}$, using K, ciphertext and given tag, decrypts the ciphertext to plaintext and verifies whether the given tag matches. If it matches then it outputs the plaintext otherwise it outputs a special symbol (say INVALID or $\perp$) to signify that the authentication failed.

Normally, nonce is used in AE schemes to make the encryption scheme probabilistic and to avoid forgery. If nonce is used by the encryption scheme then we assume that the adversary doesn't

re-use the nonce for encryption.

2.3 Modes of Authenticated Encryption

2.3.1 Basic combinations

When Authenticated Encryption was introduced in [1], the first thought towards AE was to combine Encryption and Authentication in different combinations and there are three ways namely, *a*) Encrypt-and-Authenticate; *b*) Authenticate-then-Encrypt and *c*) Encrypt-then-Authenticate. Schemes which don't depend on a particular primitives are known as *modes*. These three modes are explained along with its security.

An efficient Encryption scheme $\Pi_e(\mathcal{K}_e, \mathcal{E}_{k_e}, \mathcal{D}_{k_e})$ and an efficient MAC scheme $\Pi_t(\mathcal{K}_t, \mathcal{T}_{k_t}, \mathcal{V}_{k_t})$ is chosen. By efficient scheme, we mean a scheme which runs in time polynomial to the length of its input.

Encrypt-and-Authenticate

Let $\Pi_{eaa}(\mathcal{K}_{eaa}, \mathcal{E}_{eaa}, \mathcal{D}_{eaa})$ be the described mode. $\mathcal{K}_{eaa}$ generates two independent random keys k_e, k_t. $\mathcal{E}_{eaa}$ is encryption algorithm which is described below

$$\bar{\mathcal{E}}_{eaa}(k_e||k_t, M) = \mathcal{E}_{k_e}(M)||\mathcal{T}_{k_t}(M) = C||T$$

In this mode, both the encryption and authentication is done independently on the plaintext. Decryption is analogous and easy to understand.

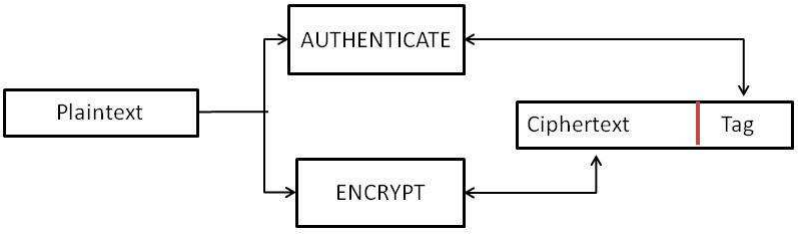

Figure 2.1: Encrypt-and-Authenticate

This mode of AE is not secure. An example showing the insecurity of this mode is available in [1].

Authenticate-then-Encrypt

Let $\Pi_{ate}(\mathcal{K}_{ate}, \mathcal{E}_{ate}, \mathcal{D}_{ate})$ be the described mode. $\mathcal{K}_{ate}$ generates two independent random keys k_e, k_t. $\mathcal{E}_{ate}$ is encryption algorithm which is described below

$$\bar{\mathcal{E}}_{ate}(k_e||k_t, M) = \mathcal{E}_{k_e}(M||\mathcal{T}_{k_t}(M)) = C$$

In this mode, first the message M is authenticated then the tag T is concatenated with the

plaintext M and then entire string $M||T$ is encrypted. For Decryption, first the ciphertext C is decrypted to get $M||T$, then Tag T is extracted and validated using algorithm $\mathcal{V}$

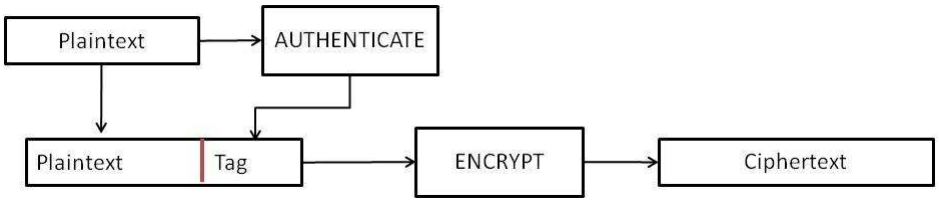

Figure 2.2: Authenticate-then-Encrypt

This mode of AE is also not secure. An example showning the weakness of this mode is available in [1].

Encrypt-then-Authenticate

Let $\Pi_{eta}(\mathcal{K}_{eta}, \mathcal{E}_{eta}, \mathcal{D}_{eta})$ be the described mode. $\mathcal{K}_{eta}$ generates two independent random keys k_e, k_t. $\mathcal{E}_{eta}$ is encryption algorithm which is described below

$$\bar{\mathcal{E}}_{eta}(k_e||k_t, M) = C||\mathcal{T}_{k_t}(C) \text{ where } C = \mathcal{E}_{k_e}(M)$$

In this mode, first the plaintext M is encrypted to get ciphertext C and this ciphertext is used as input to create the tag. For Decryption, first the ciphertext is verified and then it is decrypted to get the plaintext.

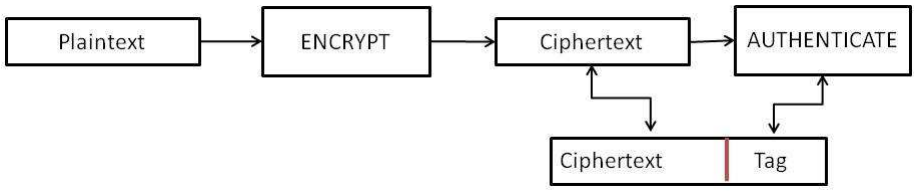

Figure 2.3: Encrypt-then-Authenticate

This mode of AE is proven secure in [1]. Since $\mathcal{E}$ is secure, the ciphertext cannot be manipulated and this ciphertext is given as input to $\mathcal{T}$.

2.3.2 Integrity Aware Parallelizable Mode

IAPM mode [7] developed by Jutla is a single pass fully parallelizable AE scheme. The underlying primitive used in IAPM is a block cipher. It requires $m + 1$ block cipher calls on plaintext of length m blocks. This is the first concrete AE mode after the introduction of AE notion by Bellare et. al and it is also provably secure. That is, if the underlying block cipher is secure then IAPM is secure; or in other words, if IAPM can be broken, then the underlying block cipher can also be broken.

The main drawback of this scheme is that it needs 2 keys and it is patented. IAPM doesn't support AEAD. Many of these drawbacks are overcome in the OCB mode described below.

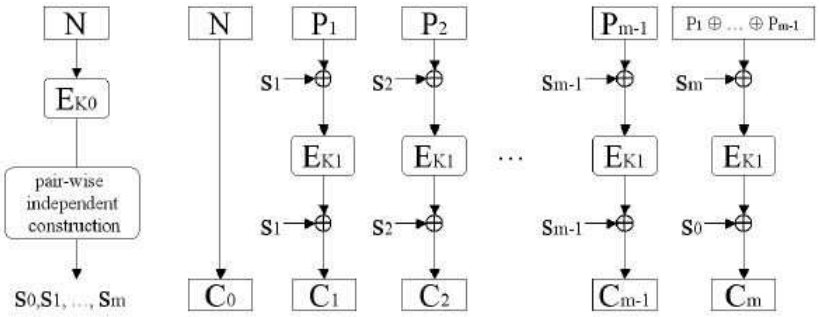

Figure 2.4: IAPM Mode

2.3.3 Offset CodeBook Mode

Offset CodeBook Mode (OCB) [15] developed by Rogaway et. al is one of the efficient AE mode both in software as well as hardware. It is inspired from the IAPM mode. OCB mode is based on block ciphers and it requires $m + 2$ block cipher invocation on plaintext of length m blocks. It is provably secure and it also supports AEAD.

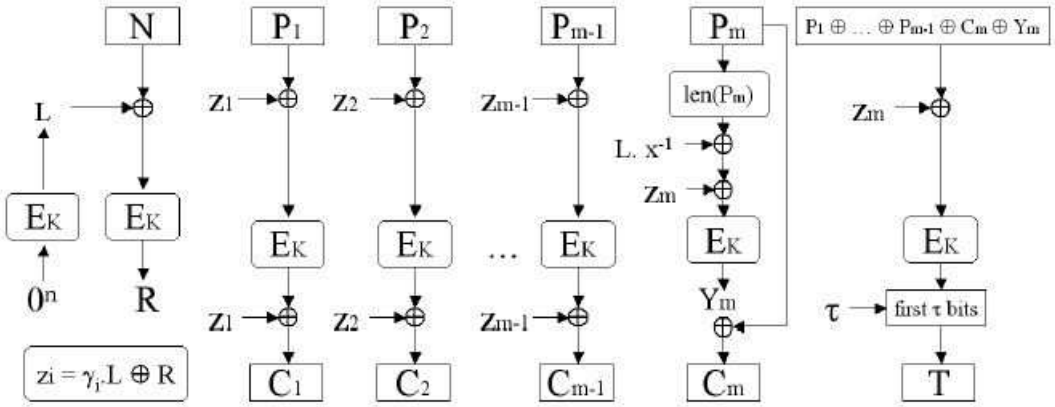

Figure 2.5: OCB Mode

The only drawback of OCB mode is that it is patented. Even though OCB mode is free to be used in open source projects but still there are few clauses which makes in not suitable for commercial purposes and other uses which are not mentioned in the license.

2.3.4 SpongeWrap

SpongeWrap [4] mode is based on the Sponge construction and developed by Bertoni et. al. This is the first AE mode which is based on permutation. The security of this construction is based on the security of Sponge construction which is provably secure, thus making SpongeWrap

a secure AE mode. It supports AEAD and it can also give intermediate tag for the plaintext if required. Further, variable length tag is also possible in this scheme. It requires m permutation calls for an m block plaintext, including associated Data. SpongeWrap can be used for key wrapping (sending the cryptographic keys through open channel).

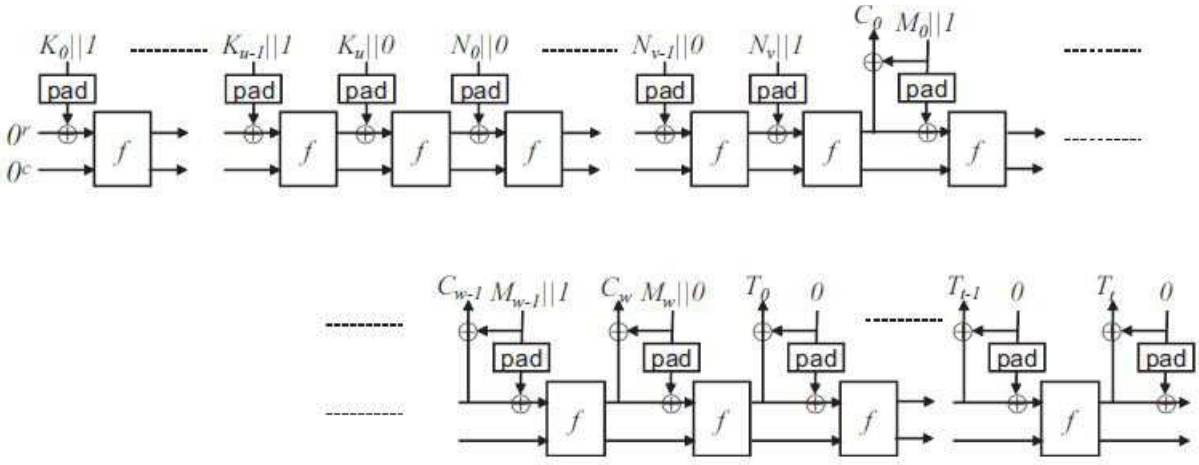

Figure 2.6: SpongeWrap

Chapter 3

Preliminaries

3.1 Provable Security

The modern notion of provable security of cryptographic schemes was first introduced by Goldwasser and Micali in [6]. Before describing provable security, we provide a brief explanation for the terms 'protocols' and 'primitives'. Primitives are building blocks in cryptography. They cannot perform useful task alone. They must be well utilized to perform a useful task and this is called protocol. For example block ciphers, hash function, random permutation are primitives and authenticated encryption, key exchange etc. are protocols.

Reduction proofs as utilized in the area of provable security provide a proof for the protocol's security based on the assumption that the underlying primitive is secure. Thus the security of the protocol is reduced to the security of the underlying primitive. The reason to follow this approach is that, there are many very good primitives like AES, SHA-2, RSA but there are many protocols which has been developed based on these have been broken because these protocols where not properly designed. Thus provable security comes handy by providing a reduction proof.

The basic method to apply the provable security paradigm is as follows. First we define the adversary model and what events lead to the success of the adversary. We also define the limitations of the adversary. Once these are defined, we try to apply these settings to the protocol and prove that if the protocol is broken then the underlying primitive can also be broken.

3.2 Adversary Models

Security proof of any protocols starts with defining the adversary model under which the protocol is secure. The assumptions of underlying primitives, the adversary's power and its limitation are defined in these models. There are three main models available. *a*) Standard Model; *b*) Random Oracle Model and *c*) Ideal Cipher Model.

3.2.1 Standard Model

This is the most commonly used model. In standard model the assumptions are based on well studied problems like factoring the product of two large primes, discrete logarithmic problem etc. The adversary can do anything within polynomial time of the length of its input. It is almost impossible to give a computational lower bound for the underlying primitives, thus it makes hard to prove many protocols under Standard Model. Since it is difficult to find suitable bound on the primitive, there is an alternate approach to solve this issue and this lead to Random Oracle Model.

3.2.2 Random Oracle Model

Bellare and Rogaway formally introduced Random Oracle Model in [2]. It is a widely used model to prove the security of the protocols. In this model, there is a public random function which takes $\{0,1\}^*$ and outputs n bits and this output is uniform and independent of other outputs. The public function is called *Random Oracle*, which we formmally define in Section 3.3 later. This model separates the primitive from the protocol for proving the security. The primitives are considered secure and then the protocols are analyzed.

3.2.3 Ideal Cipher Model

Block ciphers are widely used primitives for many protocols. Here we assume that the given block cipher is a Pseudo-Random Permutation (PRP). That is an n-bit block cipher under a chosen secret key is indistinguishable from a randomly chosen n-bit permutation.

The ideal cipher model is similar to random oracle model with certain excetpions: [5]

1. Ideal cipher need to be PRP where as a random oracle is a random function.
2. Adversaries interacting in Ideal Cipher Model have access to the cipher and its inverse whereas in Random Oracle Model adversary doesn't get access to the inverse of the random function.
3. Ideal cipher can take n bit input only whereas random oracle can take infinite bits as input.

3.3 Random Oracle

A Random Oracle (simply RO) is a public, randomly-chosen function $\mathcal{R}$ which takes an input x of infinite length and outputs $\mathcal{R}(x)$ of a fixed length. The operation of getting the output from RO is called "querying the oracle" where x is the query. Since the oracle is public, anybody can query it but nobody can evaluate the function $\mathcal{R}(\cdot)$ without querying it. RO is consistent, that

is, for a given x it will always produce the same output $\mathcal{R}(x)$ even if x is repeated any number of times.

RO is written as $\mathcal{R} : \{0,1\}^* \rightarrow \{0,1\}^n$. Let this RO be queried for inputs X_1, X_2, ... One of the key properties of an RO is that the output to the ith query is independent of the answers it has produced for earlier queries. That is, for $Y_k \in \{0,1\}^n$ for any k and $X_i \neq X_j : j < i$, we have that

$$Pr[(R(X_i) = Y_i)|(R(X_1) = Y_1) \wedge (R(X_2) = Y_2) \wedge \ldots \wedge (R(X_{i-1}) = Y_{i-1})] = 2^{-n}.$$

There are two main types of Random Oracles: Fixed-Input-Length (FIL) random oracle, whose input size is fixed and Variable-Input-Length (VIL) random oracle whose input size is not fixed.

3.4 Ideal Permutation

An ideal cipher can also be said as an ideal permutation. An ideal permutation π is a bijective function on a finite domain D and finite range R, both of which are equal sets and are chosen uniformly at random from all the available permutations. Let D and R be $\{0,1\}^{2n}$, then $\pi \xleftarrow{\$} Perm(D, D)$, where $Perm(D, D)$ is the collection of all permutations on D.

Mathematically, $\pi : D \rightarrow D$ is a permutation, if for every $y \in D$ there is one and only one $x \in D$ such that $\pi(x) = y$.

3.5 Security Notions for Authenticated Encryption

The two main security notions of the AE is privacy and authenticity of ciphertext and tag. These are the basic requirement for any AE schemes.

3.5.1 Privacy

The privacy of the encryption scheme can be proved by showing the proposed scheme is indistinguishable from a Random Oracle. The security proof is given based on the advantage of an adversary to distinguish between the Encryption oracle and the Random Oracle. This kind of proof was introduced in [15]. Consider an adversary $\mathcal{A}$ who is made to interact with either the real encryption oracle $\mathcal{E}_K(N, M)$ or an ideal function $\$(N, M)$.

If the underlying primitive is ideal random function, say ro, then ro is made available to the adversary $\mathcal{A}$ in both the cases.

If the underlying primitive is ideal permutation, then in either of these two exclusive cases, an ideal permutation π and its inverse permutation π^{-1} is available to $\mathcal{A}$.

The advantage of the adversary is the ability to identify with which it is interacting. This

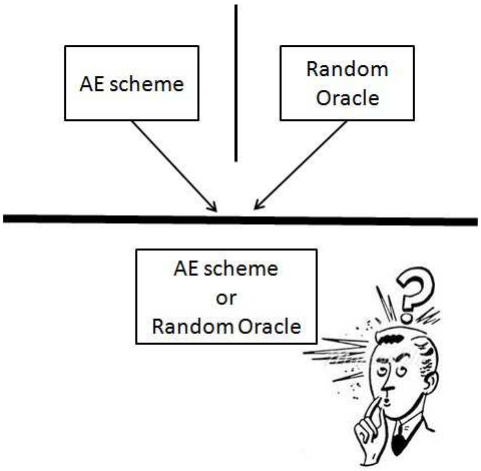

Figure 3.1: Privacy:Indistinguishability. Adversary has to find out whether it is interacting with AE scheme or a random oracle.

$$\begin{aligned} \text{For ideal permutation, } \mathbf{Adv}_{\Pi}^{priv}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathcal{E}_K(.,.),\pi,\pi^{-1}} = 1] - \Pr[\mathcal{A}^{\$(.,.),\pi,\pi^{-1}} = 1] \\ \text{For ideal random function, } \mathbf{Adv}_{\Pi}^{priv}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathcal{E}_K(.,.),ro} = 1] - \Pr[\mathcal{A}^{\$(.,.),ro} = 1] \end{aligned}$$

If the scheme uses Nonce, then the adversary $\mathcal{A}$ is nonce-respecting, i.e. it is not allowed to repeat the nonce. That is, if $\mathcal{A}$ asks the oracle query(N, M) then it will never ask the oracle with another query$(N, M^{'})$ where $M \neq M'$.

3.5.2 Authenticity

The authenticity of an AE scheme is defined in terms of the ability of an adversary who can produce a valid (N, C, T) pair without querying the oracle for that corresponding M.

The following experiment $Exp_{\Pi}^{auth}(\mathcal{A})$ is carried out to provide a bound for authenticity. The forging adversary $\mathcal{A}$ is given complete access to encryption oracle $\mathcal{E}_K(\cdot,\cdot)$, decryption oracle $\mathcal{D}_K(\cdot,\cdot,\cdot)$,either ro or (π and π^{-1}). Further, $\mathcal{A}$ can query these oracles for some fixed number of times. $\mathcal{A}$ must be nonce-respecting with encryption queries. Finally, it must output (N, C, T). We say $\mathcal{A}$ forges the AE scheme if for the given (N, C, T), the decryption algorithm outputs a valid M and the tuple (N, C, T) has not already been output by $\mathcal{E}_K()$.

The advantage of the Adversary $\mathcal{A}$ in forging the scheme Π is represented as

$$Adv_{\Pi}^{auth}(\mathcal{A}) = \Pr[Exp_{\Pi}^{auth}(\mathcal{A}) = 1].$$

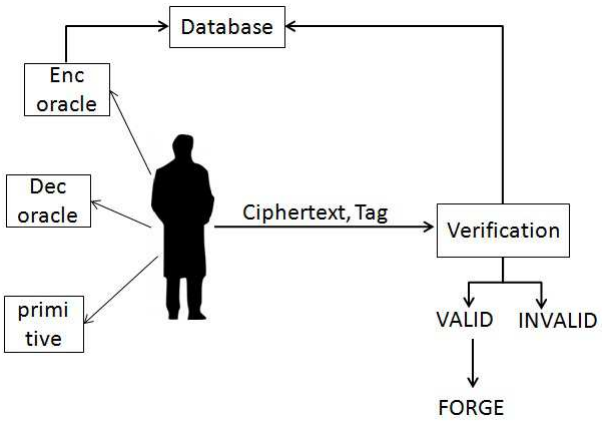

Figure 3.2: Authenticity:forgery. Adversary has to forge ciphertext, tag pair. Adversary has access to encryption, decryption and primitive oracles to forge.

3.6 Frameworks

Privacy and authenticity is proved using certain tools or in other words it is called *Framework*. In our work we used Game Playing Framework to prove the privacy of the scheme and authenticity is proved using Indiiferentaiblity Framework.

3.6.1 Game Playing Framework

Game playing framework is a technique proposed in [3]. The main purpose of this framework is to find out the ability of an adversary to distinguish an encryption scheme and a random function. Using this framework, we start with the definition of the proposed scheme and modify the scheme little by little till the scheme uses a random function. While modifying the games, the probability difference between the two games are calculated and finally all the probabilities are summed up to give the advantage of the adversary to distinguish the proposed scheme and the random function.

$$Adv_{\Pi}^{priv}(\mathcal{A}) = |\Pr[\mathcal{A}^{\text{Initial Game}} = 1] - \Pr[\mathcal{A}^{\text{Final Game}} = 1]|$$

Games are like a program. It follows pseudo code language. A game consist of three procedures *a*) Initialize; *b*) Finalize and *c*) Named Oracles (each one a procedure). The adversary can make a call to all the oracles. All the variables in the games are global and not visible to the adversary. All the variables used by adversary is local. For detailed understanding of this framework, refer [3].

3.6.2 Indifferentiability Framework

The framework was proposed by Maurer et al. in 2004 [10]. Indifferentiability is one way to show that the given scheme is secure in the hash domain.

Under an indifferentiability framework, a given scheme (say Π) with access to an ideal primitive F is $(t_A, t_S, q, \sigma, \varepsilon)$-indifferentiable from an ideal primitive G if there exists a simulator S such that for any distinguisher $\mathcal{A}$ the following equation is satisfied:

$$Adv_{\Pi}^{ind}(\mathcal{A}) = |\Pr[\mathcal{A}^{\Pi,F} = 1] - \Pr[\mathcal{A}^{G,S} = 1]| \leq \varepsilon.$$

The simulator S is an interactive algorithm which has oracle access to G and runs at most t_S times. The distinguisher $\mathcal{A}$ runs at most t_A times and makes at most q queries. The total message blocks queried by $\mathcal{A}$ is at most σ.

Chapter 4

FWPAE

4.1 Description

FWPAE is an Authenticated Encryption mode based of Fast Wide Pipe developed by M.Nandi and S.Paul [13]. To the best of the knowledge this is the first AE mode based on random function. FWPAE is nonce based AE scheme and it supports AEAD. FWPAE adds an extra functionality for FWP hash mode, thus making FWP a versatile mode of operation.

FWPAE can be used for messages of length up to 2^{64} bits. Fixed input length (FIL) random oracle (ro) is used as a primitive. The padding rule $pad(M)$ is defined as: append t zero bits and a 64-bit encoding of $|M|$ to the message M where t is the least possible integer such that $|M| + t + n + 64 = 0 \ mod \ \frac{3n}{2}$. This is same as FWP mode's padding rule.

4.2 FWP Mode

Fast Wide Pipe (FWP) is a new hash mode of operation. The main advantage of FWP is the speed of the operation. It has higher throughput compared to the Wide-Pipe mode or Sponge mode of operation. FWP uses a fixed input length random oracle which takes $(m + 2n)$ input bits and output $2n$ bits. FWP is faster because $(m + n)$ messages bits and n chaining input bits are used instead of m message bits and $2n$ chaining input bits. FWP resists Joux's multi-collision attacks. FWP mode is collision-resistance-preserving and indifferentiable from a random oracle. Security of FWP had been given to be $\frac{2n}{3}$ bit, that is FWP is secured beyond birthday bound [12]. If the proof is correct then FWP has higher security bound compared to Sponge construction. Performance wise FWP will be much better than Sponge construction for the given security bound. Thus, FWP proves to be a promising mode of operation.

4.3 Notations

Symbol	Description
$\oslash$	data is split into two equal halves
$\mathbf{E}_{\Pi}^{ro}$	FWPAE Encryption Algorithm
$\mathbf{D}_{\Pi}^{ro}$	FWPAE Decryption Algorithm
VIL	Variable Input Length

Table 4.1: Notations used

4.4 Structure

The FWPAE encryption is shown in Figure 4.1 below. The symbol $\oslash$ means that the data is split into two equal halves. That is if the data size is n bits, then once the data passes through $\oslash$, then two $n/2$ bits are produced. We assume that the size of the data passed through $\oslash$ is a multiple of 2.

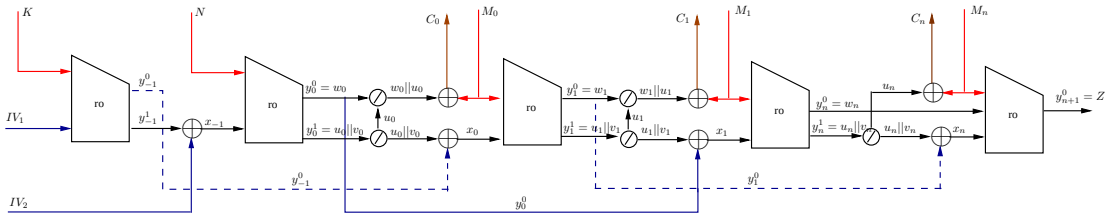

Figure 4.1: FWPAE:The symbol $\oslash$ represents splitting of bits into two equal parts. The explanation and the size of other symbol is given in Table 4.2

Description	Key	Nonce	Initial Vector 1	Initial Vector 2	Plain text	Plain text	Cipher text
Symbol	K	N	IV_1	IV_2	$m_0, .., m_{n-1}$	m_n	$c_0, .., c_{n-1}$
Size	$\frac{3n}{2}$	$\frac{3n}{2}$	n	n	$\frac{3n}{2}$	$\frac{n}{2}$	$\frac{3n}{2}$
Description	Cipher text	RO output - 1^{st} part	RO output - 2^{nd} part	RO input - 2^{nd} part	y_1-part 1	y_1–part 2	y_0
Symbol	c_n	y_0^i	y_1^i	x_i	u_i	v_i	w_i
Size	$\frac{n}{2}$	n	n	n	$\frac{n}{2}$	$\frac{n}{2}$	n

Table 4.2: Description and size of symbols used in Fig 4.1. RO stands for random oracle.

4.5 Encryption Algorithm

The algorithm for encryption,$\mathbf{E}_{\Pi}^{ro}(,,)$ is defined in Algorithm 1. The algorithm uses *ro* and *pad*, where *pad* is the padding rule as defined earlier. The input to the algorithm is the secret key K of size $\frac{3n}{2}$, nonce N of size $\frac{3n}{2}$ and message M of size at most 2^{64}. We use two initialization vectors IV_1 and IV_2 each of which is initialized to 0^n. The second initialization vector IV_2 is used for chaining.

First of all, the message M is padded using $pad(M)$. The padded message is then divided into $l(M)$ blocks. All the blocks except the last one are of size $\frac{3n}{2}$ and the last block is of size $\frac{n}{2}$. The string $K||IV_1$ is given as input to ro and the output is $y^0_{-1}||y^1_{-1}$. Then IV_2 is XORed with y^1_{-1} to form x_{-1}. The next input to the ro is $N||x_{-1}$ and we get $y^0_0||y^1_0$ as the output. At this stage, we are ready for encryption of $pad(M)$.

Let $y^1_0 = w_0$ and y^1_0 is split into two equal halves say $u_0\|v_0$. Then $w_0\|u_0$ is XORed with M_0 to give C_0 and y^1_0 is XORed with y^0_{-1} to form x_0. Now $M_0\|x_0$ becomes the input and the process continues till $l(M)-1$ block. The final block is of size $\frac{n}{2}$. Note that $C_l(M)$ is formed by XORing $M_l(M)$ with $u_{l(M)}$. The last input to the ro is $M_{l(M)}||y^0_{l(M)}||x_{l(M)}$ and output is $y^0_{l(M)+1}||y^1_{l(M)+1}$. Ciphertext of the padded message is produced as $C = C_0||C_1||...||C_{l(M)}$ and the tag produced is $T = y^0_{l(M)+1}$. The output of $\mathbf{E}^{ro}_{\Pi}(,,) = (C,T)$.

Algorithm 1: $\mathbf{E}^{ro}_{\Pi}(K,N,M)$

Input: Key K,Nonce N, Message M
Output: CipherText C, Tag T

`Initialize:` $IV_1 = IV_2^{'} = 0^n$.
$pad(M) = M_0||M_1||...||M_{l(M)}$ where $|M_i| = \frac{3n}{2},\ 0 \le i < l(M),\ |M_{l(M)}| = \frac{n}{2}$; where $l(M)$ is total number of blocks of padded message.
$(y^0_{-1}, y^1_{-1}) = ro(K \oplus IV_1)$
$x_{-1} = y^1_{-1} \oplus IV_2$
$(y^0_0, y^1_0) = ro(N||x_{-1})$
for $i = 0$ ***to*** $l(M) - 1$ **do**
    $y^0_i = w_i$
    $y^1_i = u_i||v_i$
    $c_i = M_i \oplus (w_0||u_0)$
    $x_i = y^0_i \oplus y^0_{i-1}$
    $(y^0_{i+1}, y^1_{i+1}) = ro(M_i||x_i)$
$y^0_{l(M)} = w_{l(M)}$
$y^1_{l(M)} = u_{l(M)}||v_{l(M)}$
$c_{l(M)} = M_{l(M)} \oplus u_{l(M)}$
$x_{l(M)} = y^1_{l(M)} \oplus y^1_{l(M)-1}$
$(y^0_{l(M)+1}, y^1_{l(M)+1}) = ro(M_{l(M)}||y^0_{l(M)}||x_{l(M)})$
$C = C_0||C_1||...||C_{l(M)}$
$T = y_{l(M)+1}$
return (C,T)

4.6 Decryption Algorithm

The algorithm for decryption,$\mathbf{D}^{ro}_{\Pi}(,,)$ is defined in Algorithm 2. The decryption process is similar to the encryption. All the steps are same except for XORing with the message. The only difference is that rather than XORing with M_i (which is not available), we XOR with C_i to get M_i. Once the message block M_i is evaluated we use that as input just like in the encryption.

Finally $M, T^{'}$ is evaluated and as defined, $\mathbf{D}^{ro}_{\Pi}(K, N, C, T) = (M, T^{'})$. Once the message M and tag $T^{'}$ become available, the given tag T is compared with the tag $T^{'}$. If $T^{'} = T$ then the algorithm outputs M else it output $\perp$ (INVALID) and discards the message M.

Algorithm 2: $\mathbf{D}^{ro}_{\Pi}(K, N, C, T)$

Input: Key K, Nonce N, CipherText C, Tag T
Output: Message M if T is valid, otherwise INVALID

`Initialize:` $IV_1 = IV_2 = 0^n$
$C = C_0||C_1||...||C_{l(M)}$ where $|C_i| = \frac{3n}{2}$, $0 \leq i < l(M)$. $|C_{l(M)}| = \frac{n}{2}$
$(y^0_{-1}, y^1_{-1}) = ro(K \oplus IV_1)$
$x_{-1} = y^1_{-1} \oplus IV_2$
$(y^0_0, y^1_0) = ro(N||x_{-1})$
for $i = 0$ ***to*** *l(M)-1* **do**
    $y^0_i = w_i$
    $y^1_i = u_i||v_i$
    $M_i = C_i \oplus (w_0||u_0)$
    $x_i = y^0_i \oplus y^0_{i-1}$
    $(y^0_{i+1}, y^1_{i+1}) = ro(M_i||x_i)$
$y^0_{l(M)} = w_{l(M)}$
$y^1_{l(M)} = u_{l(M)}||v_{l(M)}$
$M_{l(M)} = C_{l(M)} \oplus u_{l(M)}$
$x_{l(M)} = y^1_{l(M)} \oplus y^1_{l(M)-1}$
$(y^0_{l(M)+1}, y^1_{l(M)+1}) = ro(M_{l(M)}||y^0_{l(M)}||x_{l(M)})$
$M = M_0||M_1||...||M_{l(M)}$
$T^{'} = y_{l(M)+1}$
if $T^{'} = T$ **then**
    return (M)
else
    return $\perp$

4.7 Security Proofs

In FWPAE, we use nonce, and the adversary has the power to choose the nonce by himself but he is not allowed to repeat the nonce (we took the adversary to be nonce respecting). This restriction of not repeating the nonce is only for the encryption queries. This type of scheme is called nonce-using symmetric encryption scheme.

4.7.1 Privacy

Theorem 1. *Let FWPAE be the proposed Authenticated Encryption scheme with the defined padding rule (pad) and a fixed input length random oracle (ro) which takes $\frac{3n}{2}$ bits as input and outputs 2n bits. The advantage of the adversary A to differentiate $\mathbf{E}^{ro}_{\Pi}(,,)$ from an ideal function*

$\$()$ *is given by*

$$\boldsymbol{Adv}^{priv}_{FWPAE}(A) = Pr[A^{E_K(.,.),ro} = 1] - Pr[A^{\$(.,.),ro} = 1] \leq \frac{\sigma(\sigma+1)}{2^{\frac{n}{2}+1}},$$

where n is the bit size of tag, σ is the maximum number of queries to ro.

Proof. We have to find out the advantage of the adversary in differentiating between the outputs of E_K and random strings.

To find out the probability difference between the proposed scheme and the VIL random oracle, we use game playing framework proposed by Bellare in [3]. Using this technique we define eight games and the probability difference between two consecutive games are found. Game G0 is same as our scheme and G7 is the same as the VIL random oracle. In order to keep the proof compact, we describe these games in details after this proof.

$$\begin{aligned}
Adv^{priv}_{FWPAE}(A) &= Pr[A^{\mathbf{E}^{ro}_{\Pi},ro} = 1] - Pr[A^{\$(.,.),ro} = 1] \\
&= Pr[A^{G0} = 1] - Pr[A^{G7} = 1] \\
&= (Pr[A^{G0} = 1] - Pr[A^{G1} = 1]) + (Pr[A^{G1} = 1] - Pr[A^{G2} = 1]) \\
&\quad +(Pr[A^{G2} = 1] - Pr[A^{G3} = 1]) + (Pr[A^{G3} = 1] - Pr[A^{G4} = 1]) \\
&\quad +(Pr[A^{G4} = 1] - Pr[A^{G5} = 1]) + (Pr[A^{G5} = 1] - Pr[A^{G6} = 1]) \\
&\quad +(Pr[A^{G6} = 1] - Pr[A^{G7} = 1]) \\
&\quad \leq 0 + \frac{\sigma^2}{2^{\frac{n}{2}+1}} + \frac{\sigma}{2^{\frac{3n}{2}}} + \frac{\sigma}{2^{\frac{3n}{2}}} + \frac{\sigma}{2^{2n}} + 0 + 0 + 0 \\
&\quad \leq \frac{\sigma^2}{2^{\frac{n}{2}+1}} + \frac{\sigma}{2^{\frac{3n}{2}-1}} + \frac{\sigma}{2^{2n}} \\
&\quad \leq \frac{\sigma(\sigma+1)}{2^{\frac{n}{2}+1}}.
\end{aligned}$$

This completes the proof of Theorem 1.

□

G0 perfectly simulates ($\mathbf{E}^{ro}_{\Pi}$,ro). $\mathbf{E}^{ro}_{\Pi}$ is defined in O_1 oracle and ro is defined in O_2 oracle. First let us explain O_2 oracle and then O_1 oracle. In O_2, for a query m, it first checks whether that message is already queried. If yes, then it returns the corresponding output else it randomly outputs $2n$ bits (say $y^0_i||y^1_i$) and adds the (message, output) pair to the set X. Query to oracle O_1 uses the oracle O_2. Key, nonce and message are given as input to O_1. The working of O_1 is same as $\mathbf{E}^{ro}_{\Pi}$ and it outputs ciphertext and tag for the given (message, nonce). Thus,

$$\Pr[A^{\mathbf{E}^{ro}_{\Pi},ro} = 1] = \Pr[A^{G0} = 1].$$

Game G1: Game G0 is identical to G1 from adversary point of view. There is no difference in

the output.

$$\Pr[A^{G0} = 1] = \Pr[A^{G1} = 1]$$

Game G2: Game G2 is identical to G1 until bad. The event 'bad' is defined later. Oracle O_1 of Game G2 uses oracle O_2. The input to a query to O_2 is message concatenated with the XORed value of y_{i-1}^0 and y_i^1. Now we wish to have a random ciphertext. To make the ciphertext random, the input to the ro must always be different so that the output is always random because if the input is repeated then the output is also repeated (due to oracle O_2). From the output of ro, only $\frac{n}{2}$ bits are unknown by the adversary, remaining $\frac{3n}{2}$ bits can be seen or guessed if the adversary has some plaintext-ciphertext pairs. So, the game can control only $\frac{n}{2}$ bits. Thus in the game we make sure that $\frac{n}{2}$ bits are always different so that the output is random. If the $\frac{n}{2}$ bits have already occurred then we say that the event 'bad' has occurred and we set the variable 'bad' as true. To make these $\frac{n}{2}$ bits different, we change the previous output of ro and store the changed (message,output) pairs in a set W. The adversary is allowed to query O_2 oracle separately. If the adversary by chance queries the modified (message, output) pair then he can distinguish the oracle. When this event occurs, 'bad' is set to true. Set W stores the message-output pairs which have been modified. The adversary can also distinguish G1 and G2 if he/she guessses the key correctly. Supppose the adversary guesses the key correctly and using O_2 gets the ciphertext and then if he asks the same message to O_1, he will get a different ciphertext. Thus, he/she can distinguish both.

$$\Pr[A^{G2} = 1] - \Pr[A^{G1} = 1] = \Pr[bad \leftarrow true] + \Pr[correct\ key\ guess].$$

The probability of 'bad' to occur is based on collision in the last $\frac{n}{2}$ bits. So, to calculate the probability of 'bad', we must calculate the probability of all the collision events in various queries. Let us consider $coll_i$ means no collision happens up to $i-1$ queries and the i^{th} query produces a collision. It is clear that the probability of collision events is the union of all $coll_i$. Let $\sigma_{O_1}, \sigma_{O_2}$ stand for the total number of queries to O_1, O_2, respectively. Then $\sigma = \sigma_{O_1} + \sigma_{O_2}$

$$\begin{aligned}
\Pr[correct\ key\ guess] &\leq \frac{\sigma}{2^{\frac{3n}{2}}} \\
\Pr[bad \leftarrow true] &= \Pr[coll = 1] \\
&= \Pr[coll_1 \vee coll_2 \vee \ldots \vee coll_\sigma] \\
&\leq \Pr[coll_1] + \Pr[coll_2] + \ldots + \Pr[coll_\sigma] \\
&= \frac{1}{2^{n/2}} + \frac{2}{2^{n/2}} + \ldots + \frac{\sigma - 1}{2^{n/2}} \\
&= \frac{\sigma(\sigma-1)}{2^{\frac{n}{2}+1}} \\
&\leq \frac{\sigma^2}{2^{\frac{n}{2}+1}}. \\
\implies \Pr[A^{G1} = 1] - \Pr[A^{G2} = 1] &\leq \frac{\sigma^2}{2^{\frac{n}{2}+1}} + \frac{\sigma}{2^{\frac{3n}{2}}}.
\end{aligned}$$

Game G3: Game G2 and G3 are identical from adversary point of view until the adversary doesn't know the key. In G2 we were made sure that the random string is given as output from O_2. In G3 random string is assigned directly as output. If the adversary guesses the key correctly then he can differentiate G2 and G3 because, G2 depends on O_2 but G3 is independent of O_2. Therfore, the probability difference between the two games depends on the key guess. Let σ be the maximum number of queries to O_1 and O_2.

$$\Pr[A^{G2} = 1] - \Pr[A^{G3} = 1] = \Pr[\textit{correct key guess}] \leq \frac{\sigma}{2^{\frac{3n}{2}}}.$$

Game G4: Games G3 and G4 are identical until 'bad'. Event 'bad' is defined as the event when adversary queries O_2 with any of the messages which have earlier been used in querying O_1. Those messages are collected in set Z.

$$\Pr[A^{G3} = 1] - \Pr[A^{G4} = 1] \leq \frac{\sigma}{2^{2n}}.$$

Game G5: Games G5 and G4 are identical from the point of view of the adversary. Since random strings and XOR operations are used in them, both games are identical.

$$\Pr[A^{G5} = 1] = \Pr[A^{G4} = 1].$$

Game G6: In Game G6, the message is divided into blocks and for each message block we choose random strings. In G5, from the $2n$ random bits, only $3n/2$ bits are XORed with the message and that makes the ciphertext random. Thus the ciphertext in both the games are random from the point of view of the adversary.

$$\Pr[A^{G6} = 1] = \Pr[A^{G5} = 1].$$

G7 perfectly simulates $\$(,.,),ro$: Game G7 is the ideal case. Message is taken as input and is padded so that the output cannot be trivially distinguished based on the length of the ciphertext. After padding, a random string of length $|pad(M)|$ is assigned as ciphertext and n bit random string for tag. Oracles O_1 and O_2 are independent of each other. Therefore G7 and G6 are identical from the point of view of the adversary.

$$\Pr[A^{G7} = 1] = \Pr[A^{G6} = 1] = \Pr[A^{\$(,.,),ro} = 1].$$

4.7.2 Authenticity

The authenticity of FWPAE can be proved using the indifferentiability of FWP. Suppose we have a nonce-respecting adversary A who can forge FWPAE, then we will create a differentiable adversary B_A to differentiate FWP. Adversary B_A will have access to either (FWP, ro) or (RO, S). Similarly adversary A needs oracles $\mathbf{E}^{ro}_{\Pi}$, $\mathbf{D}^{ro}_{\Pi}, ro$ to forge. Since B_A will try to

differentiate FWP using A, thus B_A must simulate the responses for A's queries, i.e it must simulate $\mathbf{E}^{ro}_{\Pi}, \mathbf{D}^{ro}_{\Pi}, ro$ oracles.

We say that A forges FWPAE if $Exp^{auth}_{FWPAE}(A) = 1$.

Theorem 2 (Authenticity)**.** *FWPAE is forgeable with the probability*

$$Pr[Exp^{auth}_{FWPAE}(A) = 1] \leq \frac{\sigma^2 + q}{2^{n-1}},$$

where, σ is the maximum number of padded message blocks queries by A, q is the maximum number of queries to FIL ro.

Proof. Let us assume we are given a nonce-respecting adversary A who can forge FWPAE. We will create an indifferentiability adversary B_A who can differentiate (FWP, ro) from (RO, S) where S is the simulator defined in Algorithm 3.

Next we explain how B_A utilizes A to distinguish FWP.

$B_A^{O_1,O_2}$ uses O_1 and O_2 to simulate oracles $\mathbf{E}^{ro}_{\Pi}, \mathbf{D}^{ro}_{\Pi}$ and ro.

Initialize: $B_A^{O_1,O_2}$ selects a key $K \leftarrow \{0,1\}^{\frac{3n}{2}}$, where n is the tag size. Once K is chosen, it is used in $\mathbf{E}^{ro}_{\Pi}$ and $\mathbf{D}^{ro}_{\Pi}$ queries. It creates a set Q which stores all the (Nonce-Ciphertext-Tag) tuples that it sends to A.

1. Call the adversary A.
2. If A asks for $\mathbf{E}^{ro}_{\Pi}$ query by sending N, M to $B_A^{O_1,O_2}$.
   - $B_A^{O_1,O_2}$ has K, N, M. It follows the algorithm $\mathbf{E}^{ro}_{\Pi}$ with a minor change. Instead of ro, the adversary $B_A^{O_1,O_2}$ uses oracle O_2 and forms the ciphertext C. Here, the last innovation of oracle ro in algorithm $\mathbf{E}^{ro}_{\Pi}$ is ignored. For the creation of tag T, adversary $B_A^{O_1,O_2}$ sends the full message after padding to oracle O_1. Adversary $B_A^{O_1,O_2}$ returns (C, T) to A and adds N, C, T to the set Q. That is $(Q = Q \cup \{N, C, T\})$.
3. If A asks for $\mathbf{D}^{ro}_{\Pi}$ query by sending N, C, T to $B_A^{O_1,O_2}$.
   - $B_A^{O_1,O_2}$ has K, N, C, T.
   - It follows the algorithm $\mathbf{E}^{ro}_{\Pi}$ with a minor change. Instead of ro, $B_A^{O_1,O_2}$ uses oracle O_2 and obtains plaintext M. Here, the last innovation of ro oracle in $\mathbf{D}^{ro}_{\Pi}$ algorithm is ignored. Adversary $B_A^{O_1,O_2}$ sends the formed plaintext M to oracle O_1 and gets tag T'.
   - If $T = T'$ and $\{N, C, T\} \notin Q$, then $B_A^{O_1,O_2}$ returns M to A and 1 to the called function and stops, else it returns $\perp$ to A.
4. If A asks for ro query by sending m to $B_A^{O_1,O_2}$.
   - $B_A^{O_1,O_2}$ send m to O_2 and sends the reply back to A.
5. Step 2, 3, 4 can be repeated q_e, q_d, q_A times respectively.

6. If A aborts then $B_A^{O_1,O_2}$ returns 0 and stops.

Claim 1. *According to the experiment described above, the probability of the adversary distinguishing FWP from random oracle is same as the probability of the adversary forging FWPAE.*

$$Pr[B_A^{FWP,ro} = 1] = Pr[Exp_{FWPAE}^{auth}(A) = 1].$$

Before finding out the advantage of $B_A^{O_1,O_2}$ we first define the simulator S.

Simulator S [13]

FWP scheme uses a fixed input length random oracle, say, ro. To prove the indistinguishability of FWP algorithm from the distinguisher, the author created simulator S which simulates the ro used by the scheme. S takes $(m + n)$ bits of input and outputs $2n$ bits. A_{short} is an array used by the simulator and it stores all the (input-response) pairs of queries given to S or ro. The queries to S or ro are called short queries. x is a short query. MsgRecon() takes A_{short} and x and it tries to reconstruct the possible messages in which the last block of the message is x. If MsgRecon() returns any message M then it means x is the last block of that message, so the simulator sends the whole message M to VIL random oracle RO and returns the reply received from RO, if MSgCon() doesn't return any message then $ro^{'}$ is used to get the response, where $ro^{'}$ is a function which takes $(m + n)$ bits as input, and outputs $2n$ uniformly random bits. S implements $ro^{'}$. The algorithm of S is given below.

Algorithm 3: The Simulator S(.), described in [13]

Input: x: short query of length $m + n$, where m is message block size, n is tag size

Output: v: $2n$ bits string

1 $Z = \text{MsgRecon}(A_{short}, x)$, where Z is a set of possible messages which has x as the last block.
2 **if** $|Z| = 1$ **then**
3 **return** $v = RO(M)$ /* $Z = \{M\}$*/
4 **else**
5 **return** $v = ro^{'}(x)$

Now, let us find the advantage of the adversary B_A.

$$\begin{aligned} Adv_{FWP}^{ind}(B_A) &= Pr[B_A^{FWP,ro} = 1] - Pr[B_A^{RO,S} = 1] \\ &= \Pr[Exp_{FWPAE}^{auth}(A) = 1] - \Pr[B_A^{RO,S} = 1] \text{ by Claim 1.} \end{aligned}$$

From Lemma 1 and Lemma 2, we can substitute as given below.

$$\Pr[Exp_{FWPAE}^{auth}(A) = 1] = Adv_{FWP}^{ind}(B_A) + \Pr[B_A^{RO,S} = 1] \leq \frac{\sigma^2}{2^{n-1}} + \frac{q}{2^{n-1}} \leq \frac{\sigma^2 + q}{2^{n-1}}.$$

Thus, proving Theorem 2. □

Lemma 1. *[13]: The FWP hash is $(t_A, t_S, q, \sigma, \frac{\sigma^2}{2^{n-1}})$-indifferentiable in the random oracle model for the compression function, for any t_A, with $t_S = l.O(q^2)$, where t_A, t_S are the time taken by the adversary A and the simulator S, [Algorithm 6]. l is the size of message block, q and σ are the maximum number of queries and blocks used by adversary, ro is FIL random oracle, FWP is FWP mode of operation and RO is VIL random oracle.*

$$Adv^{ind}_{FWP}(A) = Pr[A^{FWP,ro} = 1] - Pr[A^{RO,S} = 1] \leq \frac{\sigma^2}{2^{n-1}}$$

Lemma 1. The detailed proof is given in [13]. □

Lemma 2. *The probability of any distinguishable adversary B_A to output 1 depends completely on whether the forgery adversary A forges FWPAE. Suppose B_A is interacting with VIL random oracle RO, simulator S and A forges FWPAE, then*

$$Pr[B_A^{RO,S} = 1] \leq \frac{q}{2^{n-1}},$$

where q is the maximum number of queries to FIL ro, q_e is the maximum number of queries to ro from encryption oracle, q_d is the maximum number of queries to ro from decryption oracle, q_A is the maximum number of queries to ro from the forging adversary A and $q = q_e + q_d + q_A$.

Lemma 2. Since B_A uses A to distinguish FWP, the only way to output 1 is that, A must forge FWPAE. That is A must output a valid tuple (N, C, T). Since B_A is interacting with RO, S, the only way A can output the valid tuple is either it can guess the tag or it can guess the key used by B_A.

$$\Pr[B_A^{RO,S} = 1] = \Pr[\text{Tag guess is correct}] + \Pr[\text{Key guess is correct}]$$

To guess the Tag correctly, the only way is to query decryption oracle. Since, B_A is interacting with RO, S, the output will always be random. Therefore, the only way to guess the Tag correctly is to query the decryption oracle. Similarly the only way to guess the key correctly, is to either query encryption oracle or the decryption oracle or S.

Let q be the maximum number of queries to FIL ro, q_e be the maximum number of queries to ro from encryption oracle, q_d be the maximum number of queries to ro from decryption oracle, q_A be the maximum number of queries to ro from the forging adversary A and $q = q_e + q_d + q_A$.

$$\Pr[\text{Correct Tag guess}] \leq \frac{q_d}{2^n} \leq \frac{q}{2^n}, \text{ and } \Pr[\text{Correct Key guess}] \leq \frac{q}{2^{3n/2}}.$$

$$\implies \Pr[B_A^{RO,S} = 1] \leq \frac{q}{2^n} + \frac{q}{2^{3n/2}} \leq \frac{q}{2^{n-1}}.$$

Thus Lemma 2 is proved. □

4.8 Games

Game G0	
1000	**Initialize:**
1001	$X = \emptyset, K \xleftarrow{\$} \{0,1\}^{\frac{3n}{2}}$
1002	$IV_1 = IV_2^{'} = 0^n$
1003	$(y_{-1}^0, y_{-1}^1) = O_2(K \| IV_1)$
1004	$x_{-1} = y_{-1}^1 \oplus IV_2$
1005	*On* $O_1 - query$ $\{N.M\}$,
1006	$pad(M) = M_0 \| M_1 \| ... \| M_\sigma$ where $\|M_i\| = \frac{3n}{2}$ $0 \leq i < \sigma$ *and* $\|M_\sigma\| = \frac{n}{2}$
1007	$(y_0^0, y_0^1) = O_2(N \| x_{-1})$
1008	**for** $i = 0$ **to** $\sigma - 1$
1009	$y_i^0 = p_i \| q_i$
1010	$y_i^1 = r_i \| s_i$
1011	$C_i = M_i \oplus (y_i^0 \| r_i)$
1012	$x_i = y_{i-1}^0 \oplus y_i^1$
1013	$w_i = M_i \| x_i$
1014	$(y_{i+1}^0, y_{i+1}^1) = O_2(w_i)$
1015	$y_\sigma^0 = p_\sigma \| q_\sigma$
1016	$y_\sigma^1 = r_\sigma \| s_\sigma$
1017	$c_\sigma = M_\sigma \oplus y_n^0$
1018	$x_\sigma = y_\sigma^1 \oplus y_{\sigma-1}^1$
1019	$w_\sigma = M_\sigma \| y_\sigma^0 \| x_\sigma$
1020	$(y_{\sigma+1}^0, y_{\sigma+1}^1) = O_2(w_\sigma)$
1021	$C = c_0 \| c_1 \| ... \| c_\sigma$
1022	$T = y_{\sigma+1}$
1023	**return** (C, T)
1024	On O_2-query m,
1025	if $(m, v) \in$ X, then **return** v
1026	$v \xleftarrow{\$} \{0,1\}^{2n}$
1027	$X = X \cup \{(m, v)\}$
1028	**return** v

Table 4.3: FWPAE: Game G0

Game G1 G2	
2000	**Initialize:**
2001	$X = \emptyset, Y = \emptyset, W = \emptyset, K \xleftarrow{\$} \{0,1\}^{\frac{3n}{2}}$
2002	$IV_1 = IV_2^{'} = 0^n$
2003	$(y_{-1}^0, y_{-1}^1) = O_2(K \| IV_1)$
2004	$x_{-1} = y_{-1}^1 \oplus IV_2$
2005	*On* $O_1 - query$ $\{N, M\}$,
2006	$pad(M) = M_0 \| M_1 \| ... \| M_\sigma$ where $\|M_i\| = \frac{3n}{2}$ $0 \leq i < \sigma$ $\quad \|M_\sigma\| = \frac{n}{2}$
2007	$(y_0^0, y_0^1) = O_2(N \| x_{-1})$
2008	**for** $i = 0$ **to** $\sigma - 1$
2009	$y_i^0 = p_i \| q_i$, where $\|p_i\|$=3n/2 and $\|q_i\|$ = n/2
2010	$y_i^1 = r_i \| s_i$
2011	$C_i = M_i \oplus (y_i^0 \| r_i)$
2012	$t_i = q_{i-1} \oplus s_i$
2013	if $t_i \in Y$,then **bad** $\leftarrow$ **true**
2014	$t_i^{'} \xleftarrow{\$} \{0,1\}^{n/2} \setminus Y$ and $Y = Y \cup t_i^{'}$
2015	$y_i^1 = r_i \| s_i^{'}$ such that $s_i^{'} = t_i^{'} \oplus q_{i-1}$
2016	$W = W \cup (m_{i-1} \| x_{i-1}, v)$ where $v = y_i^0 \| y_i^1$
2017	else $Y = Y \cup t_i$
2018	$x_i = y_{i-1}^0 \oplus y_i^1$
2019	$(y_{i+1}^0, y_{i+1}^1) = O_2(M_i \| x_i)$
2020	$y_\sigma^0 = p_\sigma \| q_\sigma$
2021	$y_\sigma^1 = r_\sigma \| s_\sigma$
2022	$c_\sigma = M_\sigma \oplus y_\sigma^0$
2023	$t_\sigma = q_{\sigma-1} \oplus s_\sigma$
2024	if $t_\sigma \in Y$,then **bad** $\leftarrow$ **true**
2025	$t_\sigma^{'} \xleftarrow{\$} \{0,1\}^{n/2} \setminus Y$ and $Y = Y \cup t_\sigma^{'}$
2026	$y_\sigma^1 = r_\sigma \| s_\sigma^{'}$ such that $s_\sigma^{'} = t_\sigma^{'} \oplus q_{\sigma-1}$
2027	$W = W \cup (m_{\sigma-1} \| x_{\sigma-1}, v)$ where $v = y_\sigma^0 \| y_\sigma^1$
2028	else $Y = Y \cup t_\sigma$
2029	$x_\sigma = y_{\sigma-1}^0 \oplus y_\sigma^1$
2030	$(y_{\sigma+1}^0, y_{\sigma+1}^1) = O_2(M_\sigma \| x_\sigma)$
2031	$C = C_0 \| C_1 \| ... \| C_\sigma$
2032	$T = y_{\sigma+1}$
2033	**return** (C, T)
2034	On O_2-query m,
2035	if $(m, v) \in W$, then **bad** $\leftarrow$ **true** // v is not random for m
2036	if $(m, v) \in$ X, then **return** v
2037	$v \xleftarrow{\$} \{0,1\}^{2n}$
2038	$X = X \cup \{(m, v)\}$
2039	**return** v

Table 4.4: FWPAE: Game G1 & G2

Game G3 G4	
3000	**Initialize:**
3001	$X = \emptyset, Y = \emptyset, Z = \emptyset$
3002	$IV_1 = IV_2^{'} = 0^n$
3003	$(y_{-1}^0, y_{-1}^1) \xleftarrow{\$} \{0,1\}^{2n}$
3004	$x_{-1} = y_{-1}^1 \oplus IV_2$
3005	*On* $O_1 - query$ $\{N, M\}$,
3006	$pad(M) = M_0 \| M_1 \| ... \| M_\sigma$ where $\|M_i\| = \frac{3n}{2}$ $0 \leq i < \sigma$ $\quad \|M_\sigma\| = \frac{n}{2}$
3007	$(y_0^0, y_0^1) \xleftarrow{\$} \{0,1\}^{2n}$
3008	**for** $i = 0$ **to** $\sigma - 1$
3009	$y_i^0 = p_i \| q_i$, where $\|p_i\|$=3n/2 and $\|q_i\|$ = n/2
3010	$y_i^1 = r_i \| s_i$
3011	$C_i = M_i \oplus (y_i^0 \| r_i)$
3012	$x_i = y_{i-1}^0 \oplus y_i^1$
3013	$(y_{i+1}^0, y_{i+1}^1) \xleftarrow{\$} \{0,1\}^{2n}$
3014	$X = X \cup (M_i \| x_i, y_{i+1}^0, y_{i+1}^1)$ and $Z = Z \cup (M_i \| x_i)$
3015	$y_\sigma^0 = p_\sigma \| q_\sigma$
3016	$y_\sigma^1 = r_\sigma \| s_\sigma$
3017	$c_\sigma = M_\sigma \oplus y_\sigma^0$
3018	$t_\sigma = q_{\sigma-1} \oplus s_\sigma$
3019	$x_\sigma = y_{\sigma-1}^0 \oplus y_\sigma^1$
3020	$(y_{\sigma+1}^0, y_{\sigma+1}^1) \xleftarrow{\$} \{0,1\}^{2n}$
3021	$X = X \cup (M_i \| x_i, y_{i+1}^0, y_{i+1}^1)$ and $Z = Z \cup (M_\sigma \| x_\sigma)$
3022	$C = C_0 \| C_1 \| ... \| C_\sigma$
3023	$T = y_{\sigma+1}$
3024	**return** (C, T)
3025	On O_2-query m,
3026	if m ∈ Z, then $bad \leftarrow true$
3027	if $(m, v) \in$ X, then
3028	**return** v
3029	$v \xleftarrow{\$} \{0,1\}^{2n}$
3030	$X = X \cup \{(m, v)\}$
3031	**return** v

Table 4.5: FWPAE: Game G3 & G4

Game G5	
5000	**Initialize:**$X = \emptyset, Y = \emptyset$
5001	*On* $O_1 - query$ $\{N, M\}$,
5002	$pad(M) = M_0 \| \| M_1 \| \| ... \| \| M_\sigma$ where $\|M_i\| = \frac{3n}{2}$ $0 \leq i < \sigma$ $\|M_\sigma\| = \frac{n}{2}$
5003	$(y_0^0, y_0^1) \xleftarrow{\$} \{0,1\}^{2n}$
5004	**for** $i = 0$ **to** $\sigma - 1$
5005	$y_i^0 = p_i \| \| q_i$, where $\|p_i\|$=3n/2 and $\|q_i\|$ = n/2
5006	$y_i^1 = r_i \| \| s_i$
5007	$C_i = M_i \oplus (y_i^0 \| \| r_i)$
5008	$(y_{i+1}^0, y_{i+1}^1) \xleftarrow{\$} \{0,1\}^{2n}$
5009	$y_\sigma^0 = p_\sigma \| \| q_\sigma$
5010	$y_\sigma^1 = r_\sigma \| \| s_\sigma$
5011	$c_\sigma = M_\sigma \oplus r_\sigma^0$
5012	$(y_{\sigma+1}^0, y_{\sigma+1}^1) \xleftarrow{\$} \{0,1\}^{2n}$
5013	$C = C_0 \| \| C_1 \| \| ... \| \| C_\sigma$
5014	$T = y_{\sigma+1}$
5015	**return** (C, T)
5016	On O_2-query m,
5017	if $(m, v) \in$ X, then **return** v
5018	$v \xleftarrow{\$} \{0,1\}^{2n}$
5019	$X = X \cup \{(m, v)\}$
5020	**return** v

Table 4.6: FWPAE: Game G5

Game G6	
6000	**Initialize:**$X = \emptyset, Y = \emptyset$
6001	*On* $O_1 - query$ $\{N, M\}$,
6002	$pad(M) = M_0 \| \| M_1 \| \| ... \| \| M_\sigma$ where $\|M_\sigma\| = \frac{3n}{2}$ $0 \leq i < \sigma$ $\|M_\sigma\| = \frac{n}{2}$
6003	**for** $i = 0$ **to** $\sigma - 1$
6004	$C_\sigma \xleftarrow{\$} \{0,1\}^{\frac{3n}{2}}$
6005	$C_n \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$
6006	$C = C_0 \| \| C_1 \| \| ... \| \| C_\sigma$
6007	$T \xleftarrow{\$} \{0,1\}^n$
6008	**return** (C, T)
6009	On O_2-query m,
6010	if $(m, v) \in$ X, then **return** v
6011	$v \xleftarrow{\$} \{0,1\}^{2n}$
6012	$X = X \cup \{(m, v)\}$
6013	**return** v

Table 4.7: FWPAE: Game G6

Game G7	
7000	**Initialize:**$X = \emptyset, Y = \emptyset$
7001	$On\ O_1 - query\ \{N, M\}$,
7002	$pad(M) = M_0 \| M_1 \| ... \| M_\sigma$ where $\|M_\sigma\| = \frac{3n}{2}\ 0 \leq i < \sigma \quad \|M_\sigma\| = \frac{n}{2}$
7003	$C \xleftarrow{\$} \{0,1\}^{\|pad(M)\|}$
7004	$T \xleftarrow{\$} \{0,1\}^n$
7005	**return** (C, T)
7006	On O_2-query m,
7007	if $(m, v) \in$ X, then **return** v
7008	$v \xleftarrow{\$} \{0,1\}^{2n}$
7009	$X = X \cup \{(m, v)\}$
7010	**return** v

Table 4.8: FWPAE: Game G7

Chapter 5

FPAE

5.1 Description

FPAE is permutation based AE mode. It is developed from FP [14] hash mode of operation. FPAE is nonce based AE scheme which supports AEAD. The privacy and authenticity of FPAE shows that it has a better security comparable to SpongeWrap.

5.2 FP Mode

FP is a permutation based hash mode of operation. The design of FP mode is derived from FWP mode of operation created by M. Nandi et al. [13]. The FP mode uses an easy-to-invert permutation instead of hard-to-invert function as in the FWP. This makes the FP mode efficient, easy to implement and fast. In [13], it is shown that the FP mode is indifferentiable from a random oracle up to $2^{n/2}$ queries. Hence this mode resists all generic attacks including multi-collision attack, 2^{nd} pre-image attack and herding attack. The permutation used in this mode is based on only one assumption: that there is no structural weakness. In the paper, the authors claim that it is possible to extend the security bound of indifferentiablility with a random oracle to a value close to n bits. If this is indeed proven then the FP mode of operation will have better security compared to the Sponge construction which provides only $n/2$ bit security.

5.3 Structure

The encryption of FPAE is shown in Fig. 5.1 next. FPAE is a permutation based Authenticated Encryption scheme and this permutation is represented by π. Key K is of size n bits. Nonce N is public and is n bits long. $M_1 \dots M_p$ is a padded message where $\forall i \ |M_i| = n$ bits. Similarly $C = C_1 \dots C_p$ is a ciphertext of size which is a multiple of n bits. Finally, Tag T is also of size n bits.

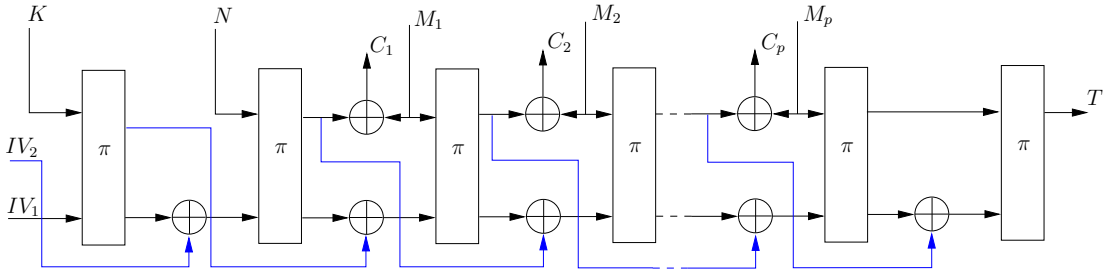

Figure 5.1: The proposed authenticated encryption scheme FPAE. Symbols used in the figure are described in Table 5.1.

Symbol	K	N	IV_1	IV_2	$M_0, \ldots, M_p$	$C_0, \ldots, C_p$	T
Meaning	Key	nonce	IV	IV	padded message	ciphertext	tag
Size (bits)	n	n	n	n	n	n	n

Table 5.1: Description and size of symbols used in Fig 4.1. RO stands for random oracle.

5.4 Encryption Algorithm

The encryption algorithm of FPAE, referred to as FPAE-E in the rest of this work, is defined in Algorithm 4. The encryption algorithm has access to a permutation π and a $Pad()$ function which produces padded message.

First let us describe the $Pad()$ function and then π. In FPAE, we use the same $Pad()$ function as used in the FP mode. $Pad(M) = M\|1\|0^t$ is the padding rule used by FP mode where, $Pad(M) = M_1\|M_2\|\ldots\|M_p$, such that $|M_i| = n$ for $1 \leq i \leq p$. Here t is the smallest non-negative integer such that $|M| + 1 + t = 0 \bmod n$.

The permutation π used in FPAE takes a string of $2n$ bits (say $x\|x'$) as input and produces a $2n$ bit string as output (say $y\|y'$). Now we are ready to describe the encryption algorithm.

The FPAE-E algorithm takes Key K, nonce N and the message M as input and produces ciphertext C and Tag T as the output for the given N, M. FPAE uses a $2n$ bit IV, which is split into two halves, called IV_1 and IV_2, each n bits long. Both these $IV^{'}s$ are initialized to all zero strings 0^n. Algorithm 4 along with Fig. 5.1 provides the complete description of the encryption operation. The algorithm returns the cipheretxt and tag pair C, T.

5.5 Decryption Algorithm

The decryption algorithm FPAE-D is defined in algorithm 5. The decryption algorithm has access to the same π which was used in the encryption algorithm. The algorithm takes Key K, nonce N, Ciphertext C and tag $T^{'}$ as input and decrypts the ciphertext to get message M and tag T. If $T = T^{'}$ then it outputs M else it outputs $\perp$. The algorithm is described next.

Algorithm 4: FPAE-E$^{\pi,Pad}(K, N, M)$

Input: Key K, Nonce N, Message M
Output: CipherText C, Tag T

`Initialize:` $IV_1 = IV_2 = 0^n$.
$Pad(M) = M_1||M_1||...||M_p$ *where* $\forall i : |M_i| = n$
$x_{-1} = IV_1, x'_{-1} = K$
$y_0 \| y'_0 = \pi(x_{-1} \| x'_{-1})$
$x_0 = y_0 \oplus IV_2$
$x'_0 = N$
$y_1 \| y'_1 = \pi(x_0 \| x'_0)$
for $i = 1$ ***to*** p **do**
    $C_i = y'_i \oplus M_i$
    $x_i = y_i \oplus y'_{i-1}$
    $x'_i = M_i$
    $y_{i+1} \| y'_{i+1} = \pi(x_i \| x'_i)$
$x_{p+1} = y_{p+1} \oplus y'_p$
$x'_{p+1} = y'_{p+1}$
$y_{p+2} \| y'_{p+2} = \pi(x_{p+1} \| x'_{p+1})$
$C = C_1 \| C_1 \| \ldots \| C_p$
$T = y'_{p+2}$
return (C, T)

5.6 Security Proofs

In FPAE, we use nonce, and the adversary has the power to choose the nonce by himself but he is not allowed to repeat the nonce (we took the adversary to be nonce respecting). This restriction of not repeating the nonce is only for the encryption queries. This type of scheme is called nonce-using symmetric encryption scheme.

5.6.1 Privacy

Theorem 3. *Let $FPAE^{\pi,Pad}$ be the proposed Authenticated Encryption scheme with the defined padding rule (Pad) and ideal permutation (π) which operates on $2n$ bits. The adversary $\mathcal{A}$ is given acces to π, π^{-1} and the advantage of $\mathcal{A}$ to differentiate FPAE-$E^{\pi,Pad}$ from an ideal function $\$()$ is given by*

$$\begin{aligned} \boldsymbol{Adv}^{priv}_{FPAE}(\mathcal{A}) &= Pr[\mathcal{A}^{E_K,\pi,\pi^{-1}} = 1] - Pr[\mathcal{A}^{\$(.,.),\pi,\pi^{-1}} = 1] \\ &\leq \frac{q(2\sigma+1)}{2^n} + \frac{5\sigma(\sigma-1)}{2^{2n+1}}, \end{aligned}$$

where E_K represents FPAE-$E^{\pi,Pad}$, n is the size of the tag, σ is the total number of queries to π, π^{-1} by FPAE-$E^{\pi,Pad}$,FPAE-$D^{\pi^{-1}}$. The maximum number of queries to FPAE-$E^{\pi,Pad}$, π and π^{-1} by $\mathcal{A}$ are q_1, q_2 and q_3, respectively. Finally, $q = q_1 + q_2 + q_3$.

Algorithm 5: FPAE-D$^{\pi}(K, N, C, T^{'})$

Input: Key K, Nonce N, CipherText C, Tag $T^{'}$
Output: Message M or $\perp$

`Initialize:` $IV_1 = IV_2 = 0^n$.
$C = C_1||C_1||...||C_p$ where $\forall i : |C_i| = n$
$x_{-1} = IV_1, x'_{-1} = K$
$y_0 \| y'_0 = \pi(x_{-1} \| x'_{-1})$
$x_0 = y_0 \oplus IV_2$
$x'_0 = N$
$y_1 \| y'_1 = \pi(x_0 \| x'_0)$
for $i = 1$ ***to*** p **do**
    $M_i = y'_i \oplus C_i$
    $x_i = y_i \oplus y'_{i-1}$
    $x'_i = M_i$
    $y_{i+1} \| y'_{i+1} = \pi(x_i \| x'_i)$
$x_{p+1} = y_{p+1} \oplus y'_p$
$x'_{p+1} = y'_{p+1}$
$y_{p+2} \| y'_{p+2} = \pi(x_{p+1} \| x'_{p+1})$
$M = M_1 \| M_1 \| \dots \| M_p$
$T = y'_{p+2}$
if $T' = T$ **then**
    return M
else
    return $\perp$

Proof. The advantage of the adversary is the ability to differentiate the proposed scheme from a random oracle. We use game playing framework proposed by Bellare et al. in [3] to compute the probability difference. We define a sequence of eight games and compute the probability differences between consecutive games. Game G0 represents our proposed scheme and Game G8 represent Variable Input Length (VIL) random oracle (RO). In order to keep the proof compact, we describe these games in details later in the appendix.

Using the equations (5.1), (5.2), (5.6), (5.7), (5.10), (5.11), (5.15), (5.16) and (5.17), we find the

advantage of the adversary, as follows.

$$\begin{aligned}
Adv^{priv}_{FPAE}(\mathcal{A}) &= Pr[\mathcal{A}^{E_K,\pi,\pi^{-1}} = 1] \\
&\quad -Pr[\mathcal{A}^{\$(,.,),\pi,\pi^{-1}} = 1] \\
&= Pr[\mathcal{A}^{G0} = 1] - Pr[\mathcal{A}^{G8} = 1] \\
&= (Pr[\mathcal{A}^{G0} = 1] - Pr[\mathcal{A}^{G1} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G1} = 1] - Pr[\mathcal{A}^{G2} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G2} = 1] - Pr[\mathcal{A}^{G3} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G3} = 1] - Pr[\mathcal{A}^{G4} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G4} = 1] - Pr[\mathcal{A}^{G5} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G5} = 1] - Pr[\mathcal{A}^{G6} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G6} = 1] - Pr[\mathcal{A}^{G7} = 1]) \\
&\quad +(Pr[\mathcal{A}^{G7} = 1] - Pr[\mathcal{A}^{G8} = 1]) \\
&\leq 0 + \frac{\sigma(\sigma-1)}{2^{2n}} + 0 + \frac{\sigma(\sigma-1)}{2^{2n+1}} \\
&\quad +\frac{q}{2^n} + \frac{\sigma(\sigma-1)}{2^{2n}} + \frac{q\sigma}{2^{n-1}} \\
&\quad +0+0 \\
&\leq \frac{q\sigma}{2^{n-1}} + \frac{q}{2^n} + \frac{5\sigma(\sigma-1)}{2^{2n+1}} \\
&\leq \frac{q(2\sigma+1)}{2^n} + \frac{5\sigma(\sigma-1)}{2^{2n+1}}.
\end{aligned}$$

This completes the proof of Theorem 3. □

G0 perfectly simulates (FPAE,π, π^{-1}): Oracles O_2 and O_3 perfectly simulate an ideal permutation π and its inverse π^{-1}. Oracle O_1 uniformly selects a random key K and simulates FPAE-E$^{\pi,Pad}(K,.,.)$ using O_2 as π. Set X stores all the input-output pairs of oracle O_2. G0 is defined in Table 5.2.

$$\Pr[\mathcal{A}^{E_K,\pi,\pi^{-1}} = 1] = \Pr[\mathcal{A}^{G0} = 1]. \tag{5.1}$$

Game G1: Game G0 is identical to G1. This is clear from the definitions of games G0 and G1 (defined in Table 5.3).

$$\Pr[\mathcal{A}^{G0} = 1] = \Pr[\mathcal{A}^{G1} = 1]. \tag{5.2}$$

Game G2: Game G1 and G2 are identical until *bad*. Thus the adversary can differentiate G1 and G2 only when *bad* occurs.

$$\Pr[\mathcal{A}^{G1} = 1] - \Pr[\mathcal{A}^{G2} = 1] = \Pr[bad \leftarrow true] \tag{5.3}$$

Bad occurs when the input received by O_2 and O_3 collides with the elements in the set X. Let $Pr[\text{coll} = 1]$ be the probability of occurrence of *bad* in O_2 and O_3. Since, the set X is used by

both the oracles, the probability is same for both.

$$\Pr[bad \leftarrow true] = 2\Pr[\text{coll} = 1]. \tag{5.4}$$

Let $Pr[coll_i]$ means that there is no collision till $i-1$ queries and collision occurs in the i^{th} query. Thus, $Pr[\text{coll} = 1]$ is given by:

$$\begin{aligned}
&\Pr[coll = 1] \\
&= \Pr[coll_1 \vee coll_2 \vee \ldots \vee coll_\sigma] \\
&\leq \Pr[coll_1] + \Pr[coll_2] + \ldots + \Pr[coll_\sigma] \\
&\leq \frac{1}{2^{2n}} + \frac{2}{2^{2n}} + \ldots + \frac{\sigma-1}{2^{2n}} \\
&\leq \frac{\sigma(\sigma-1)}{2^{2n+1}}.
\end{aligned} \tag{5.5}$$

Thus from equations (5.3), (5.4) and (5.5) the probability difference between G1 and G2 is:

$$\Pr[\mathcal{A}^{G1} = 1] - \Pr[\mathcal{A}^{G2} = 1] \leq \frac{\sigma(\sigma-1)}{2^{2n}}. \tag{5.6}$$

Game G3: Games G2 and G3 are identical. In G3, oracle O_1 is independent of oracle O_2, but from the adversarial point of view, both the games are identical.

$$\Pr[\mathcal{A}^{G2} = 1] = \Pr[\mathcal{A}^{G3} = 1]. \tag{5.7}$$

Game G4: Games G3 and G4 can be differentiated by *bad* events and by key guessing.

$$\begin{aligned}
&\Pr[\mathcal{A}^{G3} = 1] - \Pr[\mathcal{A}^{G4} = 1] \\
&= \Pr[\text{bad events}] + \Pr[\text{correct key guess}].
\end{aligned}$$

We first discuss about the *bad* events. The *bad* events are collision events as discussed earlier. Thus the probability calculations are similar to the previous games.

$$\Pr[\text{bad events}] \leq \frac{\sigma(\sigma-1)}{2^{2n+1}}. \tag{5.8}$$

Key guessing can also distinguish between games G3 and G4. Suppose the adversary guesses the key and runs oracle O_2 on (N, M) and re-runs with the same data on O_1. Then the output can be differentiated.

$$\Pr[\text{correct key guess}] \leq \frac{q}{2^n}. \tag{5.9}$$

From equations (5.8) and (5.9) the probability difference between G3 and G4 is:

$$\Pr[\mathcal{A}^{G3} = 1] - \Pr[\mathcal{A}^{G4} = 1] \leq \frac{\sigma(\sigma-1)}{2^{2n+1}} + \frac{q}{2^n}. \tag{5.10}$$

Game G5: Game G4 and G5 are identical in oracle O_1 queries but both the games can be differentiated by oracles O_2 and/or O_3 queries. In G4, oracles O_2 and O_3 implement ideal random functions, whereas in G5 both the oracles implements ideal permutation. Thus both these games are identical as long as there is no collision in queries to O_2 and O_3. The probability calculation is therefore similar to the one in game G2.

$$\Pr[\mathcal{A}^{G4} = 1] - \Pr[\mathcal{A}^{G5} = 1] \leq \frac{\sigma(\sigma - 1)}{2^{2n}}. \tag{5.11}$$

Game G6: Adversary can differentiate games G5 and G6 based on the occurrence of *bad* event.

$$\Pr[\mathcal{A}^{G5} = 1] - \Pr[\mathcal{A}^{G6} = 1] = \Pr[\text{bad event}]$$

Let the sets W and Y store all the $2n$ bit inputs and $2n$ bit outputs for oracle O_1 respectively. According to the game definition [Table 5.5], the $2n$ bit output is randomly selected. Out of these $2n$ bits, adversary knows n bits using message, ciphertext and tag. Hence, only the remaining n bits are random and secret. The *bad* event occurs when the adversary queries O_2 with 'm' which is already present in the set W or the adversary queries O_3 with 'v' which is present in Y. Set W and Y can have at most σ elements. An element of W is represented as W_i and an element of Y by Y_j where $1 \leq i \leq \sigma$ and $1 \leq j \leq \sigma$. The adversary can query O_2 and O_3 at most q_2 and q_3 times, respectively.

Therefore if we have $M_i = W_j$ (where $1 \leq i \leq q_2$ and $1 \leq j \leq \sigma$) then it is a *bad* event. Similarly, if we have $V_i = Y_j$ (where $1 \leq i \leq q_3$ and $1 \leq j \leq \sigma$), then it is a *bad* event.

$$\Pr[\text{bad event}] = \sum_{i=1}^{q_2} \sum_{j=1}^{\sigma} \Pr[m_i = W_j] + \sum_{i=1}^{q_3} \sum_{j=1}^{\sigma} \Pr[v_i = Y_j]. \tag{5.12}$$

The summation of probability is the maximum number of collisions within the sets W and Y respectively.

$$\sum_{i=1}^{q_2} \sum_{j=1}^{\sigma} \Pr[m_i = W_j] \leq \frac{q_2\sigma}{2^n} \leq \frac{q\sigma}{2^n}. \tag{5.13}$$

$$\sum_{i=1}^{q_3} \sum_{j=1}^{\sigma} \Pr[v_i = Y_j] \leq \frac{q_3\sigma}{2^n} \leq \frac{q\sigma}{2^n}. \tag{5.14}$$

Thus from equations (5.12), (5.13) and (5.14), the probability difference between G5 and G6 is:

$$\Pr[\mathcal{A}^{G5} = 1] - \Pr[\mathcal{A}^{G6} = 1] \leq \frac{q\sigma}{2^{n-1}}. \tag{5.15}$$

Game G7: G6 and G7 are identical from adversarial point of view. In G6, we made sure that no input-output pairs are queried by the adversary to O_2 and O_3, thus making O_1 independent of other oracles as in G7.

$$\Pr[\mathcal{A}^{G6} = 1] = \Pr[\mathcal{A}^{G7} = 1]. \tag{5.16}$$

Game G8: G7 and G8 are identical from adversarial point of view. In G7, for each message block, a random string is assigned as the ciphertext, and another random string is assigned as the tag. On the other hand, random strings are assigned as ciphertext and tag for the complete (padded) message. G8 perfectly simulates $\$(), \pi, \pi^{-1}$.

$$\Pr[\mathcal{A}^{G7} = 1] = \Pr[\mathcal{A}^{G8} = 1] = \Pr[\mathcal{A}^{\$(,.),\pi,\pi^{-1}} = 1]. \tag{5.17}$$

5.6.2 Authenticity

The authenticity of an AE scheme is defined in terms of the ability of an adversary who can produce a valid (N, C, T) pair. The forging adversary $\mathcal{A}$ can query π, π^{-1}, FPAE-E$^{\pi,Pad}(K, ., .)$, FPAE-D$^{\pi}(K, ., ., .)$ oracles at most q_1, q_2, q_e, q_d times, respectively. $\mathcal{A}$ must be nonce-respecting with encryption queries. Finally, $\mathcal{A}$ outputs (N, C, T). We say $\mathcal{A}$ forges the AE scheme if the decryption algorithm outputs M and the corresponding (N, C, T) tuple has not been output by the encryption oracle.

Let $Exp^{auth}_{\text{FPAE},\pi,\pi^{-1}}(\mathcal{A})$ be the forging experiment which is defined as follows:

1. $\mathcal{A}$ can query π, π^{-1}, FPAE-E$^{\pi,Pad}(K, ., .)$ and FPAE-D$^{\pi}(K, ., ., .)$ at most q_1, q_2, q_e and q_d times, respectively.
2. All the query responses of FPAE-E$^{\pi,Pad}(K, ., .)$ are stored in a set, say Z. Set Z has elements of the type (N_i, C_i, T_i).
3. Let M be a valid message. If FPAE-D$^{\pi}(N, C, T) = M$ and $(N, C, T) \notin Z$ then output 1.
4. After all the queries, output 0.

We say that $\mathcal{A}$ forges the scheme if $Exp^{auth}_{\pi,\pi^{-1}}(\mathcal{A}) = 1$.

The advantage of the Adversary $\mathcal{A}$ in forging the scheme is represented as

$$Adv^{auth}_{\text{FPAE}}(\mathcal{A}) = \Pr[Exp^{auth}_{\text{FPAE},\pi,\pi^{-1}}(\mathcal{A}) = 1].$$

Theorem 4 (Authenticity)**.** *FPAE is forgeable with the probability*

$$Pr[Exp^{auth}_{FPAE,\pi,\pi^{-1}}(\mathcal{A}) = 1] \leq \frac{28\sigma^2}{2^n} + \frac{q}{2^{n-1}},$$

where, σ is the maximum number of queries to π, π^{-1} by $\mathcal{A}$ through FPAE-$E^{\pi,Pad}(K, ., .)$ and FPAE-$D^{\pi}(K, ., ., .)$, q is the maximum number of queries to $\pi, \pi^{-1}/S, S^{-1}$ by FPAE-$E^{\pi,Pad}(K, ., .)$, FPAE-$D^{\pi}(K, ., ., .)$ together. q_1, q_2, q_e, q_d is the maximum queries to π, π^{-1}, FPAE-$E^{\pi,Pad}(K, ., .)$, FPAE-$D^{\pi}(K, ., ., .)$ directly by $\mathcal{A}$. Let, $q = q_1 + q_2 + q_e + q_d$.

Proof: (Authenticity). We will upper bound the following probability

$$\Pr[Exp^{auth}_{\text{FPAE},\pi,\pi^{-1}}(\mathcal{A}) = 1]$$

by reducing the forging experiment to indifferentiability of FPAE.

Suppose we have a forging adversary $\mathcal{A}$ who can forge FPAE, then we will create another differentiable adversary B_A who can differentiate FP from a random oracle RO.

Adversary $\mathcal{A}$ need access to oracles (FPAE,π, π^{-1}). B_A has access to oracles (O_1, O_2, O_3) which can either be (FP, π, π^{-1}) or (RO, S, S^{-1}), where S, S^{-1} are defined in Algorithms 6 and 7 respectively. Now B_A will use its own oracle to simulate $(FPAE, \pi, \pi^{-1})$ for A.

Let us show how $B_A^{O_1,O_2,O_3}$ differentiates FP from random oracle using the forging adversary $\mathcal{A}$ for FPAE.

Initialize: $B_A^{O_1,O_2,O_3}$ selects a key $K \leftarrow \{0,1\}^n$. Once K is chosen, it is used in $FPAE$-E and $FPAE$-D queries. It creates a set Q which stores all the (Nonce-Ciphertext-Tag) tuples that it sends to $\mathcal{A}$.

1. Call the adversary $\mathcal{A}$.
2. If $\mathcal{A}$ asks for FPAE-$E^{\pi,Pad}$ query by sending N, M to $B_A^{O_1,O_2,O_3}$.
   - $B_A^{O_1,O_2,O_3}$ has K, N, M.
   - It follows the algorithm FPAE-$E^{\pi,Pad}$ with a minor change. Instead of π, the adversary $B_A^{O_1,O_2,O_3}$ uses oracle O_2 and forms the ciphertext C. Here, the last invocation of oracle π in algorithm FWPAE-$E^{\pi,Pad}$ is ignored. For the creation of tag T, adversary $B_A^{O_1,O_2,O_3}$ sends the padded message to oracle O_1. Adversary $B_A^{O_1,O_2,O_3}$ returns (C, T) to $\mathcal{A}$ and adds N, C, T to the set Q. That is $Q \leftarrow Q \cup \{N, C, T\}$.
3. If $\mathcal{A}$ asks for FPAE-D^{π} query by sending N, C, T to $B_A^{O_1,O_2,O_3}$.
   - $B_A^{O_1,O_2,O_3}$ has K, N, C, T.
   - It follows the algorithm FPAE-D^{π} with a minor change. Instead of π, $B_A^{O_1,O_2,O_3}$ uses oracle O_2 and obtains plaintext M. Here, the last innovation of ro oracle in $FWPAE$-D^{π} algorithm is ignored. Adversary $B_A^{O_1,O_2,O_3}$ sends the formed plaintext M to oracle O_1 and gets tag T'.
   - If $T = T'$ and $\{N, C, T\} \notin Q$, then $B_A^{O_1,O_2,O_3}$ returns M to $\mathcal{A}$ and 1 to the called function and stops, else it returns $\perp$ to $\mathcal{A}$.
4. If $\mathcal{A}$ asks for π query by sending m to $B_A^{O_1,O_2,O_3}$.
   - $B_A^{O_1,O_2,O_3}$ sends m to O_2 and sends the reply back to $\mathcal{A}$.
5. If $\mathcal{A}$ asks for π^{-1} query by sending v to $B_A^{O_1,O_2,O_3}$.
   - $B_A^{O_1,O_2,O_3}$ sends v to O_3 and sends the reply back to $\mathcal{A}$.
6. Steps 2, 3, 4 and 5 can be queried at most q_e, q_d, q_1, q_2 times, respectively.
7. If either $\mathcal{A}$ aborts or all the allowed queries made by $\mathcal{A}$ have been exhausted (i.e. those corresponding to encryption, decryption, pi and pi^{-1} oracles), then $B_A^{O_1,O_2,O_3}$ returns 0 and stops.

Claim 2. *According to the experiment described above, the probability of the adversary distinguishing FP from random oracle is same as the probability of the adversary forging FPAE.*

$$Pr[B_A^{FP,\pi,\pi^{-1}} = 1] = Pr[Exp^{auth}_{FPAE,\pi,\pi^{-1}}(\mathcal{A}) = 1].$$

Before computing the advantage of $B_A^{O_1,O_2,O_3}$, we first define the simulator S, S^{-1}.

Simulator pair S, S^{-1} [14]

To prove the indifferentiability of FP mode, [14] created a simulator pair so that the adversary cannot differentiate FP mode from a random oracle.

Before describing the simulator, lets give a brief description of the data structure and functions used by the simulator pair. Set D_s contains all the query-response pairs of S, S^{-1}. M is reconstructible message from the set D_s, that is D_s contains all the query-response pair required to compute $FP(M)$. T_s is a graph which stores all the reconstructible messages from root to leaves. $FullGraph(D_s)$ function is used to create T_s from D_s. The function $MsgRecon(x, T_s)$ returns all the messages M such that the final input to S is x.

We describe the simulator S next. S takes a string x of $2n$ bits as input and outputs r of $2n$ bits. First, the string r is selected randomly. Then with the given input x and T_s, we check for the existence of a valid reconstructible message M. If the function $MsgRecon()$ returns a message M, then we query $RO(M)$ and modify the last n bits of r with the output received from RO and return this modified value of r. On the other hand, if $MsgRecon()$ returns an empty set, then update the set D_s, recreate T_s by calling $FullGraph(D_s)$ and return r.

Algorithm 6: The Simulator S(.), described in [14]

Input: x: short query of length $2n$
Output: r: $2n$ bits string

$r \xleftarrow{\$} \{0,1\}^{2n}$
if $r \in D_s$ **then**
  Abort
$M = \text{MsgRecon}(x, T_s)$
if $|M| = 1$ **then**
  return $r[n, 2n-1] = RO(M)$
$D_s[x] = r$
$FullGraph(D_s)$
return r

Simulator S^{-1} receives $2n$ bit string r and outputs x. The simulator selects $2n$ bit string randomly and assign that to x. If this x is alredy present in the domain of D_s, then it aborts the process else it updates D_s and recreates T_s. Finally it returns the x.

Algorithm 7: The Simulator $S^{-1}(.)$, described in [14]

Input: r: $2n$ bit string
Output: x: $2n$ bits string

$x \xleftarrow{\$} \{0,1\}^{2n}$
if $x \in D_s$ **then**
  Abort
$D_s[x] = r$
$FullGraph(D_s)$
return x

Now, let us find the advantage of the adversary B_A.

$$\begin{aligned} & Adv^{ind}_{FP}(B_A) \\ = \; & Pr[B_A^{FP,\pi,\pi^{-1}} = 1] - Pr[B_A^{RO,S,S^{-1}} = 1] \\ = \; & \Pr[Exp^{auth}_{FPAE,\pi,\pi^{-1}}(\mathcal{A}) = 1] - \Pr[B_A^{RO,S,S^{-1}} = 1] \\ & \text{-by Claim 2.} \end{aligned}$$

From Lemma 3 and Lemma 4, we can substitute as given below.

$$\begin{aligned} & \Pr[Exp^{auth}_{FPAE,\pi,\pi^{-1}}(\mathcal{A}) = 1] \\ = \; & Adv^{ind}_{FP}(B_A) + \Pr[B_A^{RO,S,S^{-1}} = 1] \\ \leq \; & \frac{28\sigma^2}{2^n} + \frac{q}{2^{n-1}} \end{aligned}$$

Thus, proving Theorem 4. □

Lemma 3. *[14]: The FP hash is $(t_A, t_S, \sigma, \frac{28\sigma^2}{2^n})$-indifferentiable from a random oracle, where $t_A = \infty, t_S = O(\sigma^5)$ and $\sigma \leq \mathrm{K}2^{n/2}$, where* K *is a fixed constant derived from ϵ, t_A, t_S are the time taken by the adversary $\mathcal{A}$ and the simulator S, S^{-1}, [Algorithm 6, 7]. σ is the maximum number of queries to π, π^{-1} by $\mathcal{A}$ through FPAE-$E^{\pi,Pad}(K,.,.)$ and FPAE-$D^{\pi}(K,.,.,.)$, π, π^{-1} is fixed permutation and its inverse, respectively, FP is FP mode of operation and RO is VIL random oracle.*[1]

$$\begin{aligned} Adv^{ind}_{FP}(B_A) &= Pr[B_A^{FP,\pi,\pi^{-1}} = 1] - Pr[B_A^{RO,S,S^{-1}} = 1] \\ &\leq \frac{28\sigma^2}{2^n}. \end{aligned}$$

Proof: The detailed proof is given in [14]. □

Lemma 4. *The probability of any differentiable adversary B_A to output* 1 *depends completely on whether the forgery adversary $\mathcal{A}$ forges FPAE. Suppose B_A is interacting with VIL random*

[1] In [14] the authors have not considered the maximum block length of each query to FP.

oracle RO, simulators S, S^{-1} and $\mathcal{A}$ forges FPAE, then

$$Pr[B_A^{RO,S,S^{-1}} = 1] \leq \frac{q}{2^{n-1}},$$

Proof: Since B_A uses $\mathcal{A}$ to distinguish FP, the only way to output 1 is that, A must forge FPAE. That is $\mathcal{A}$ must output a valid tuple (N, C, T). Since B_A is interacting with RO, S, S^{-1}, the only way $\mathcal{A}$ can output the valid tuple is either by guessing the tag or by guessing the key used by B_A.

$$\begin{aligned}\Pr[B_A^{RO,S,S^{-1}} = 1] = \Pr[&\text{Correct Tag} \wedge \text{Wrong Key}] \\ &+\Pr[\text{Correct Tag} \wedge \text{Correct Key}].\end{aligned}$$

Since B_A is interacting with RO, S, S^{-1}, the output will always be random. Therefore, the only way to get the Tag correct with wrong key is to guess the tag.

$$\Pr[\text{Correct Tag} \wedge \text{Wrong Key}] \leq \frac{q}{2^n}.$$

Similarly the only way to get the key correctly is to guess it. Adversary can use any oracle to find the key.

$$\Pr[\text{Correct Tag} \wedge \text{Correct Key}] \leq \frac{q}{2^n}$$

$$\implies \Pr[B_A^{RO,S} = 1] \leq \frac{q}{2^{n-1}}.$$

Thus Lemma 4 is proved. □

5.7 Games

Game G0	
100	**Initiaze:**
101	$X = \emptyset, K \xleftarrow{\$} \{0,1\}^n$
102	$IV_1 = IV_2 = 0^n$
103	$x_{-1} = IV_1,\ x^{'}_{-1} = K$
104	$y_0 \| {y_0}' = O_2(x_{-1} \| x^{'}_{-1})$
105	*On* $O_1 - query\ \{N, M\}$,
106	$pad(M) = M_1 \| M_1 \| ... \| M_p,$ where $\forall i : \|M_i\| = n$
107	$x_0 = y_0 \oplus IV_2,\ {x_0}' = N$
108	$y_1 \| {y_1}' = O_2(x_0 \| x^{'}_0)$
109	**for** $i = 1$ **to** p
110	$C_i = {y_i}' \oplus M_i$
111	$x_i = y_i \oplus y^{'}_{i-1},\ x^{'}_i = M_i$
112	$y_{i+1} \| {y_{i+1}}' = O_2(x_i \| x^{'}_i)$
113	$x_{p+1} = y_{p+1} \oplus y^{'}_p,\ {x_{p+1}}' = {y_{p+1}}'$
114	$y_{p+2} \| {y_{p+2}}' = O_2(x_{p+1} \| x^{'}_{p+1})$
115	$C = C_1 \| C_2 \| \ldots \| C_p$ and $T = {y_{p+2}}'$
116	**Return** (C, T)
117	On O_2-query m,
118	if $(m, v) \in X$, then **return** v
119	$v \xleftarrow{\$} \{0,1\}^{2n}$
120	if $\exists (m', v^{'}) \in X$ s.t $v' = v$ then
121	$v \xleftarrow{\$} \{0,1\}^{2n} \setminus \{v^{'} : (m', v') \in X\}$
122	$X = X \cup \{(m, v)\}$
123	**return** v
123	On O_3-query v, /*Inverse query.*/
124	if $(m, v) \in X$, then **return** m
125	$m \xleftarrow{\$} \{0,1\}^{2n}$
126	if $\exists (m', v^{'}) \in X$ s.t $m' = m$ then
127	$m \xleftarrow{\$} \{0,1\}^{2n} \setminus \{m^{'} : (m^{'}, v^{'}) \in X\}$
128	$X = X \cup \{(m, v)\}$
129	**return** m

Table 5.2: FPAE: Game G0

Game	G1	G2
200	**Initiaze:**	
201	$X = \emptyset, K \xleftarrow{\$} \{0,1\}^n$	
202	$IV_1 = IV_2 = 0^n$	
203	$x_{-1} = IV_1,\ x^{'}_{-1} = K$	
204	$y_0 \| {y_0}' = O_2(x_{-1} \| x^{'}_{-1})$	
205	*On* $O_1 - query\ \{N, M\}$,	
206	$pad(M) = M_1 \| M_1 \| ... \| M_p,$ where $\forall i : \|M_i\| = n$	
207	$x_0 = y_0 \oplus IV_2,\ {x_0}' = N$	
208	$y_1 \| {y_1}' = O_2(x_0 \| x^{'}_0)$	
209	**for** $i = 1$ **to** p	
210	$C_i = {y_i}' \oplus M_i$	
211	$x_i = y_i \oplus y^{'}_{i-1},\ x^{'}_i = M_i$	
212	$y_{i+1} \| {y_{i+1}}' = O_2(x_i \| x^{'}_i)$	
213	$x_{p+1} = y_{p+1} \oplus y^{'}_p,\ {x_{p+1}}' = {y_{p+1}}'$	
214	$y_{p+2} \| {y_{p+2}}' = O_2(x_{p+1} \| x^{'}_{p+1})$	
215	$C = C_1 \| C_2 \| \ldots \| C_p$ and $T = {y_{p+2}}'$	
216	**Return** (C, T)	
217	On O_2-query m,	
218	if $(m, v) \in X$, then **return** v	
219	$v \xleftarrow{\$} \{0,1\}^{2n}$	
220	if $\exists (m', v') \in X$ s.t $v' = v$ then **bad** $\leftarrow$ **true**	
221		$\boxed{v \xleftarrow{\$} \{0,1\}^{2n} \setminus \{v^{'} : (m^{'}, v^{'}) \in X\}}$
222	$X = X \cup \{(m, v)\}$	
223	**return** v	
223	On O_3-query v, /*Inverse query.*/	
224	if $(m, v) \in X$, then **return** m	
225	$m \xleftarrow{\$} \{0,1\}^{2n}$	
226	if $\exists (m', v') \in X$ s.t $m' = m$ then **bad** $\leftarrow$ **true**	
227		$\boxed{m \xleftarrow{\$} \{0,1\}^{2n} \setminus \{m^{'} : (m', v') \in X\}}$
228	$X = X \cup \{(m, v)\}$	
229	**return** m	

Table 5.3: FPAE: Game G1 & G2

Game	G3	G4
300	**Initiaze:**	
301	$X = \emptyset, K \xleftarrow{\$} \{0,1\}^n, IV_1 = IV_2 = 0^n$	
302	$x_{-1} = IV_1,\ x^{'}_{-1} = K$	
303	$y_0 \| {y_0}' \xleftarrow{\$} \{0,1\}^{2n}$	
304	$X = X \cup \{(x_{-1} \| x^{'}_{-1}, y_0 \| {y_0}')\}$	
305	**On O_1-query {N,M},**	
306	$pad(M) = M_1 \| M_1 \| ... \| M_p, \forall i : \|M_i\| = n$	
307	$x_0 = y_0 \oplus IV_2,\ {x_0}' = N$	
308	**if** $(x_0 \| x^{'}_0, v) \in X$ then **bad** $\leftarrow$ **true**	
309		$y_1 \| {y_1}' = v$
310	**else**	
311		$y_1 \| {y_1}' \xleftarrow{\$} \{0,1\}^{2n}$
312		$X = X \cup \{(x_0 \| x^{'}_0, y_1 \| {y_1}')\}$
313	**end if**	
314	**for** $i = 1$ **to** p	
315	$C_i = {y_i}' \oplus M_i$	
316	$x_i = y_i \oplus y^{'}_{i-1},\ x^{'}_i = M_i$	
317	**if** $(x_i \| x^{'}_i, v) \in X$ then **bad** $\leftarrow$ **true**,	
318		$y_{i+1} \| {y_{i+1}}' = v$
319	**else**	
320		$y_{i+1} \| {y_{i+1}}' \xleftarrow{\$} \{0,1\}^{2n}$
321		$X = X \cup \{(x_i \| x^{'}_i, y_{i+1} \| {y_{i+1}}')\}$
322	**end if**	
323	**end for**	
324	$x_{p+1} = y_{p+1} \oplus y^{'}_p,\ {x_{p+1}}' = {y_{p+1}}'$	
325	**if** $(x_{p+1} \| x^{'}_{p+1}, v) \in X$ then **bad** $\leftarrow$ **true**,	
326		$y_{p+2} \| {y_{p+2}}' = v$
327	**else**	
328		$y_{p+2} \| {y_{p+2}}' \xleftarrow{\$} \{0,1\}^{2n}$
329		$X = X \cup \{(x_{p+1} \| x^{'}_{p+1}, y_{p+2} \| {y_{p+2}}')\}$
330	**end if**	
331	$C = C_1 \| C_2 \| \ldots \| C_p$ and $T = {y_{p+2}}'$	
332	**Return** (C, T)	
333	On O_2-query m,	
334	if $(m, v) \in X$, then **return** v	
335	$v \xleftarrow{\$} \{0,1\}^{2n}, X = X \cup \{m, v\}$	
336	**return** v	
336	On O_3-query v, /*Inverse query.*/	
337	if $(m, v) \in X$, then **return** m	
338	$m \xleftarrow{\$} \{0,1\}^{2n}, X = X \cup \{m, v\}$	
339	**return** m	

Table 5.4: FPAE: Game G3 & G4

Game G5	G6
500	**Initiaze:**
501	$X = Y = W = \emptyset, K \xleftarrow{\$} \{0,1\}^n$
502	$IV_1 = IV_2 = 0^n$
503	$x_{-1} = IV_1,\ x_{-1}^{'} = K$
504	$y_0 \| {y_0}' \xleftarrow{\$} \{0,1\}^{2n}$
505	$X = X \cup \{(x_{-1} \| x_{-1}^{'}, y_0 \| {y_0}')\}$
506	$Y = Y \cup \{(y_0 \| {y_0}')\}, W = W \cup \{(x_{-1} \| x_{-1}^{'})\}$
507	$On\ O_1 - query\ \{N, M\},$
508	$pad(M) = M_1 \| M_1 \| ... \| M_p,\ \forall i : \|M_i\| = n$
509	$x_0 = y_0 \oplus IV_2,\ {x_0}' = N$
510	$y_1 \| {y_1}' \xleftarrow{\$} \{0,1\}^{2n}$
511	$X = X \cup \{(x_0 \| x_0^{'}, y_1 \| {y_1}')\}$
512	$Y = Y \cup \{(y_1 \| {y_1}')\}, W = W \cup \{(x_0 \| x_0^{'})\}$
513	**for** $i = 1$ **to** p
514	$C_i = {y_i}' \oplus M_i$
515	$x_i = y_i \oplus y_{i-1}^{'},\ x_i^{'} = M_i$
516	$y_{i+1} \| {y_{i+1}}' \xleftarrow{\$} \{0,1\}^{2n}$
517	$X = X \cup \{(x_i \| x_i^{'}, y_{i+1} \| {y_{i+1}}')\}$
518	$Y = Y \cup \{(y_{i+1} \| {y_{i+1}}')\}, W = W \cup \{(x_i \| x_i^{'})\}$
519	**end for**
520	$x_{p+1} = y_{p+1} \oplus y_p^{'},\ {x_{p+1}}' = {y_{p+1}}'$
521	$y_{p+2} \| {y_{p+2}}' \xleftarrow{\$} \{0,1\}^{2n}$
522	$X = X \cup \{(x_{p+1} \| x_{p+1}^{'}, y_{p+2} \| {y_{p+2}}')\}$
523	$Y = Y \cup \{(y_{p+2} \| {y_{p+2}}')\}$
524	$W = W \cup \{(x_{p+1} \| x_{p+1}^{'})\}$
525	$C = C_1 \| C_2 \| \ldots \| C_p$ and $T = {y_{p+2}}'$
526	**Return** (C, T)
527	On O_2-query m,
528	if $(m) \in W$, then **bad** $\leftarrow$ **true** ABORT
529	if $(m, v) \in X$, then **return** v
530	$v \xleftarrow{\$} \{0,1\}^{2n}$
531	if $\exists (m', v') \in X$ s.t $v' = v$ then
532	$v \xleftarrow{\$} \{0,1\}^{2n} \setminus \{v^{'} : (m', v') \in X\}$
533	$X = X \cup \{(m, v)\}$
534	**return** v
534	On O_3-query v, /*Inverse query.*/
535	if $(v) \in Y$, then **bad** $\leftarrow$ **true** ABORT
536	if $(m, v) \in X$, then **return** m
537	$m \xleftarrow{\$} \{0,1\}^{2n}$
538	if $\exists (m', v') \in X$ s.t $m' = m$ then
539	$m \xleftarrow{\$} \{0,1\}^{2n} \setminus \{m^{'} : (m^{'}, v^{'}) \in X\}$
540	$X = X \cup \{(m, v)\}$
541	**return** m

Table 5.5: FPAE: Game G5 & G6

Game G7	
700	**Initiaze:**
701	$X = \emptyset$
702	*On* $O_1 - query$ $\{N, M\}$,
703	$pad(M) = M_1 \| \| M_1 \| \| ... \| \| M_p,\ \forall i : \|M_i\| = n$
704	**for** $i = 1$ **to** p
705	$C_i \xleftarrow{\$} \{0,1\}^{2n}$
706	**end for**
707	$T \xleftarrow{\$} \{0,1\}^n$
708	$C = C_1 \| C_2 \| \ldots \| C_p$
709	**Return** (C, T)
710	On O_2-query m,
711	if $(m, v) \in X$, then **return** v
712	$v \xleftarrow{\$} \{0,1\}^{2n}$
713	if $\exists \{m', v'\} \in X$ s.t $v' = v$ then
714	$v \xleftarrow{\$} \{0,1\}^{2n} \setminus \{v^{'} : (m', v') \in X\}$
715	$X = X \cup \{(m, v)\}$
716	**return** v
716	On O_3-query v, /*Inverse query.*/
717	if $(m, v) \in X$, then **return** m
718	$m \xleftarrow{\$} \{0,1\}^{2n}$
719	if $\exists (m', v') \in X$ s.t $m' = m$ then
720	$m \xleftarrow{\$} \{0,1\}^{2n} \setminus \{m^{'} : (m^{'}, v^{'}) \in X\}$
721	$X = X \cup \{(m, v)\}$
722	**return** m

Table 5.6: FPAE: Game G7

Game G8	
800	**Initiaze:**
801	$X = \emptyset$
802	*On* $O_1 - query$ $\{N, M\}$,
803	$pad(M) = M_1 \| M_1 \| ... \| M_p,\ \forall i : \|M_i\| = n$
804	$C \xleftarrow{\$} \{0,1\}^{\|pad(M)\|}$
805	$T \xleftarrow{\$} \{0,1\}^n$
806	**Return** (C, T)
807	On O_2-query m,
808	if $(m, v) \in X$, then **return** v
809	$v \xleftarrow{\$} \{0,1\}^{2n}$
810	if $\exists \{m', v'\} \in X$ s.t $v' = v$ then
811	$v \xleftarrow{\$} \{0,1\}^{2n} \setminus \{v^{'} : (m', v') \in X\}$
812	$X = X \cup \{(m, v)\}$
813	**return** v
813	On O_3-query v, /*Inverse query.*/
814	if $(m, v) \in X$, then **return** m
815	$m \xleftarrow{\$} \{0,1\}^{2n}$
816	if $\exists (m', v') \in X$ s.t $m' = m$ then
817	$m \xleftarrow{\$} \{0,1\}^{2n} \setminus \{m^{'} : (m^{'}, v^{'}) \in X\}$
818	$X = X \cup \{(m, v)\}$
819	**return** m

Table 5.7: FPAE: Game G8

Chapter 6

Conclusion

In this thesis, we designed FWPAE and FPAE modes for authenticated encrytion and provided the proofs for privacy and authenticity of the two modes. FWPAE is the first AE mode based on random function to the best of our knowledge. FWPAE and FPAE are showing a promising security bounds, thus making these two efficient AE modes. Further, in the recently concluded SHA-3 competition, one of the selection criteria of the hash function was that the underlying primitive must support many features. Thus in this work, Authenticared Encryption has been developed based on FWP and FP hash mode and making these a more versatile hash modes. We hope that this work will generate more intetest in the community to develop modes for special purposes using different underlying primitives.

Chapter 7

Future Work

The next thing that can be done is to develop *online* Authenticated Encryption modes from FWPAE and FPAE. Online here we mean that the hardware which does the cryptographic task has limited memory. Hence online helps is encrypting in real time. Here the main concern is the decryption. Normally, in decryption, complete ciphertext is required to authenticate the data but this contradicts the purpose of online. Here the trick is to make the decryption sort of online. One direction of research can be to make the decryption sort of online.

FWPAE and FPAE are modes of operation. The next thing that can be done is to instantiate the random permutation and random function and get the exact implementation results and compare with other AE schemes. This will give us the practical efficiency of our modes. We can use different ciphers to instantiate random function and random permutation and have a comparitive study.

Bibliography

[1] Bellare, M., and Namprempre, C. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *ASIACRYPT* (2000), T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, Springer, pp. 531–545.

[2] Bellare, M., and Rogaway, P. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security* (1993), D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, Eds., ACM, pp. 62–73.

[3] Bellare, M., and Rogaway, P. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT* (2006), S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*, Springer, pp. 409–426.

[4] Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography* (2011), A. Miri and S. Vaudenay, Eds., vol. 7118 of *Lecture Notes in Computer Science*, Springer, pp. 320–337.

[5] Black, J. The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. In *FSE* (2006), M. J. B. Robshaw, Ed., vol. 4047 of *Lecture Notes in Computer Science*, Springer, pp. 328–340.

[6] Goldwasser, S., and Micali, S. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *STOC* (1982), H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, Eds., ACM, pp. 365–377.

[7] Jutla, C. S. Encryption Modes with Almost Free Message Integrity. In *EUROCRYPT* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer, pp. 529–544.

[8] Katz, J., and Lindell, Y. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007.

[9] Kohno, T., Viega, J., and Whiting, D. Cwc: A high-performance conventional authenticated encryption mode. In *FSE* (2004), B. K. Roy and W. Meier, Eds., vol. 3017 of *Lecture Notes in Computer Science*, Springer, pp. 408–426.

[10] Maurer, U. M., Renner, R., and Holenstein, C. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In *TCC* (2004), M. Naor, Ed., vol. 2951 of *Lecture Notes in Computer Science*, Springer, pp. 21–39.

[11] Mihir Bellare, P. R. Lecture Notes in Cryptography. `http://cseweb.ucsd.edu/~mihir/cse207/classnotes.html`.

[12] Moody, D., Paul, S., and Smith-Tone, D. Indifferentiability Security of the Fast Widepipe Hash: Breaking the Birthday Barrier. *IACR Cryptology ePrint Archive 2011* (2011), 630.

[13] Nandi, M., and Paul, S. Speeding Up the Wide-Pipe: Secure and Fast Hashing. In *INDOCRYPT* (2010), G. Gong and K. C. Gupta, Eds., vol. 6498 of *Lecture Notes in Computer Science*, Springer.

[14] Paul, S., Homsirikamol, E., and Gaj, K. A Novel Permutation-Based Hash Mode of Operation FP and the Hash Function SAMOSA. In *INDOCRYPT* (2012), S. D. Galbraith and M. Nandi, Eds., vol. 7668 of *Lecture Notes in Computer Science*, Springer, pp. 509–527.

[15] Rogaway, P., Bellare, M., Black, J., and Krovetz, T. Ocb: a block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security* (2001), M. K. Reiter and P. Samarati, Eds., ACM, pp. 196–205.

[16] Whiting, D., Housley, R., and Ferguson, N. Counter with CBC-MAC (CCM). *Available from* `csrc.nist.gov/encryption/modes/proposedmodes/ccm/ccm.pdf` (2003).